Utilisation et détermination d'hypergraphes de précédence pour la conception et l'équilibrage des lignes d'assemblage.

Laurent Relange

N° d'ordre : 942

Année 2002

THÈSE

préparée au

Laboratoire d'Automatique de Besançon (UMR CNRS 6596)

présentée à

L'U.F.R. DES SCIENCES ET TECHNIQUES DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

pour obtenir le

GRADE DE DOCTEUR DE L'UNIVERSITÉ

spécialité **Automatique et Informatique**

Utilisation et détermination d'hypergraphes de précédence pour la conception et l'équilibrage des lignes d'assemblage

par

Laurent Relange

(DEA *spécialité* Informatique, Automatique et Productique)

Soutenue le 16 décembre 2002 devant la commission d'examen :

Rapporteurs	**Alain Delchambre** (Professeur à l'Université Libre de Bruxelles)
	Alexandre Dolgui (Professeur à l'Université de Technologie Troyes)
Examinateurs	**Alain Haurat** (Professeur à l'Université de Savoie)
(Président de jury)	**Michel Gourgand** (Professeur à l'Université Blaise Pascal -Clermond Ferrand II)
Directeur de thèse	**Jean-Michel Henrioud** (Professeur à l'Université de Franche-Comté)
	Christophe Perrard (Maître de Conférences à l'Université de Franche-Comté)

Table des matières

Table des définitions

Table des propriétés

Table des algorithmes

Table des figures

Glossaire des notations

E	Ensemble des tâches
D	Ensemble des séquences d'enchaînement possibles
Υ	Ensemble de séquences d'enchaînement admissibles
$\bar{\Upsilon}$	Ensemble de séquences d'enchaînement non admissibles
Υ_i	Sous-ensemble quelconque de séquences d'enchaînement admissibles
x_i	Séquence d'enchaînement admissibles de l'ensemble Υ
x	Séquence d'enchaînement admissibles quelconque
a, b	Tâche d'assemblage quelconque
α	Tâche de chargement du composant de base α
α_i	Tâche d'assemblage portant sur le composant α_i
α	Tâche de déchargement du produit fini
$x_{i_{\alpha[}}$	Sous-séquence d'enchaînement de x_i de la première tâche à la tâche α exclue
$x_{i_{\alpha]}}$	Sous-séquence d'enchaînement de x_i de la première tâche à la tâche α inclue
$x_{i_{[\alpha}}$	Sous-séquence d'enchaînement de x_i de la tâche α inclue à la dernière tâche
$x_{i_{]\alpha}}$	Sous-séquence d'enchaînement de x_i de la tâche α exclue à la dernière tâche
S	Sous-assemblage
C	Constituant
$\mathcal{E}(x_i)$	État engendré avec la séquence x_i
$\mathcal{E}$	État quelconque
$\mathcal{E}_i$	État i
$\mathcal{A}$	ASTD
ξ	Ensemble des états admissibles d'un ASTD

τ	Ensemble des opérations, permettant de passer d'un état $\mathcal{E}_i$ à un état $\mathcal{E}_j$ de ξ
G_D	Graphe d'enchaînement
ω^+	Demi-cocycle supérieur
$\pounds_\Upsilon$	Expression logique de l'ensemble Υ
$\bigoplus x_i$	Disjonction exclusive généralisée sur un ensemble de séquences d'enchaînement x_i
$\sum x_i$	Disjonction généralisée sur un ensemble de séquences d'enchaînement x_i
$\prod(\alpha_i \rightarrow \alpha_{i+1})$	Conjonction généralisée sur un ensemble des précédences des tâches α_i sur les tâches α_{i+1}
ξ	Ensemble des états admissibles d'un ASTD
τ	Ensemble des opérations, permettant de passer d'un état i à un état j, définies sur $\xi \times \xi$
G_D	Graphe d'enchaînement
σ	Ensemble des successions directes des tâches réalisables de l'ensemble E, définies sur $E \times E$
L	Application de σ dans ξ

Remerciements

Avec cet ouvrage, je vous présente les résultats de mes recherches effectuées au sein de l'équipe « Conception intégrée » au *Laboratoire d'Automatique de Besançon*. Cette équipe est menée par J. M. Henrioud, Professeur à l'Université de Franche-Comté et Directeur de l'école Doctorale Sciences Physique pour l'Ingénieur et Microtechniques.

Je tiens en premier lieu à lui exprimer ici ma gratitude pour m'avoir permis d'effectuer ces travaux ; ses conseils et ses compétences m'ont aidés à parvenir au termes de mon travail.

Par ces quelques lignes, je marque ma reconnaissance à A. Bourjault, Professeur à l'Ecole Nationale Supérieure de Mécanique et des Microtechniques et Directeur du *Laboratoire d'Automatique de Besançon*, pour son accueil et les moyens qu'il m'a donné pour effectuer ces recherches.

Je remercie Messieurs A. Delchambre Professeur à l'Université Libre de Bruxelles et A. Dolgui Professeur à l'Université de Technologie Troyes, pour l'honneur qu'ils m'ont fait en acceptant d'être rapporteur de ma thèse.

Je remercie également Messieurs M. Gourgand, A. Haurat et C. Perrard, pour l'honneur qu'ils m'ont fait en acceptant de juger ces travaux de thèse.

Je tiens également à remercier tous mes collègues, ayant participé de près comme de loin à l'élaboration de ce travail.

Je tiens également à remercier Zabou et Edith, pour leur énergie dépensée, qui nous permet de travailler dans des locaux propres et agréables, et pour leur amitié.

Je tiens également à remercier tous mes amis, ayant participé à mon bonheur lors de ces travaux, et pour leur soutien.

Afin de n'oublier personne, je remercie également toutes les personnes qui ont pu quelque part m'apporter un peu de soutien.

Je remercie ma famille pour son soutien et sa compréhension.

Je remercie mon amour pour elle.

Comme un ange qui se dévoile,
tu me regardais dans ma nuit,
avec ton beau regard d'étoile
qui m'éblouit . . .

L'âme en fleur

Victor Hugo

A.

Ma famille,

Mes amis,

Mon amour,

Introduction générale

AVEC UN MONDE NOUVEAU, avec une Europe nouvelle de libre échange, avec une monnaie nouvelle qui permettent un transit des produits, matières, ressources pour ainsi dire libre, avec une concurrence de plus en plus rude, nous devons penser à une industrie nouvelle. Pour ce faire, cette dernière doit être de plus en plus compétitive et réactive. De nouvelles technologies apparaissent chaque jour, la multiplication des produits, les familles de produits, la réduction des délais, des coûts de fabrications, nous poussent à imaginer des nouvelles manières de travailler. Sachant que nous ne pouvons pas toujours construire de nouvelles usines, ou de nouveaux systèmes de production, nous devons essayer d'utiliser autrement les systèmes et industries existants. Pour cela, certains proposent de remplacer les hommes par des machines, robots, chaînes..., mais ces changements sont excessivement coûteux. De là est née l'idée d'une amélioration de l'ensemble en optimisant les systèmes existants.

L'étude de ces améliorations est connue dans le monde scientifique sous le nom de *Productique*. La Productique se compose de plusieurs axes dont deux étant l'usinage et l'assemblage. L'usinage est déjà très avancé dans le domaine de la conception et de la conduite des systèmes de production.

C'est un fait, les réseaux de Petri sont très utilisés en *gestion de flux* et en *maintenance et sûreté de fonctionnement*, les graphes de précédence sont actuellement l'outil d'aide à la conception le plus répandu. En ordonnancement, le graphe PERT est très utilisé mais il n'est en fait qu'un graphe de précédence.

L'objectif de ce travail est de définir une méthode de génération de graphes de précédence à partir des séquences d'assemblage établies par le logiciel LEGA.

Le premier chapitre de cet ouvrage sera un panorama des connaissances scientifiques en ce domaine. Nous présenterons pour commencer les systèmes de production, puis nous parlerons plus précisément de l'assemblage des produits, mais aussi des systèmes

d'assemblage. Un grand nombre d'auteurs sera cité afin de positionner notre contribution au sein de la communauté scientifique.

Dans le deuxième chapitre nous définirons les connaissances de base nécessaires pour mieux comprendre les problèmes de conception. Nous développerons de manière plus poussée deux modélisations des processus d'assemblage utilisés comme outils d'aide à la conception des systèmes d'assemblage : les graphes de précédence et les ASTD[1]. Cette présentations apportera les connaissances suffisantes pour mieux comprendre les chapitres trois et quatre.

Dans les troisième et quatrième chapitres nous exposerons deux méthodes différentes de génération de graphes de précédence. Le chapitre trois est une méthode de génération basée sur la transformation de graphes, à partir de graphes d'assemblage représentant les processus d'assemblage, jusqu'à obtention des graphes de précédence souhaités.

Le chapitre quatre propose une méthode de génération systématique des graphes de précédence basée sur la logique booléenne. Notre méthode de génération permet d'obtenir des graphes de précédence « valides » et « exhaustifs », et cette méthode peut aussi engendrer des hypergraphes de précédence.

En annexe A, nous rappellerons quelques définitions sur les graphes. Pour terminer nous parlerons dans l'annexe B de la méthode de génération des graphes de précédence de K.S. Naphade, modifiée par P. De Lit, et nous proposerons un complément à leur travaux.

[1] Assembly State Transition Diagram *ou* Graphes d'état des transitions d'assemblage de simple niveau

Chapitre 1

Les systèmes de production

Un système de production manufacturier est un système qui transforme un ou plusieurs objets en un ou plusieurs autres objets plus élaborés. Il peut être par exemple, un système d'usinage, qui représente une partie budgétaire importante dans la réalisation de certains produits (comme par exemple l'automobile) car il fait partie de la première phase de la production du produit.

Un *système d'assemblage* permet de passer d'un ensemble d'objets indépendants les uns des autres à un ou plusieurs objets par *agrégation*. Les études sur la conception des systèmes d'assemblage, faisant partie de la deuxième phase de la production, sont en *retard* par rapport aux études sur la conception d'ateliers d'usinage. Afin de mieux comprendre les *systèmes d'assemblage*, quelques notions sur ces systèmes seront développées, dans ce chapitre, puis leur *conception* sera abordée, quelques possibilités de *modélisations* de produits et de *représentations* possibles des processus d'assemblage seront présentées. Nous ferons ressortir, dans ce chapitre, la problématique de ce travail.

1.1 Système d'assemblage

L'ensemble des rappels empruntés à A. Bourjault [10], à J.M. Henrioud [30] et à K. Chen [12] de cette section portera sur l'assemblage ; les *systèmes d'assemblage* seront définis, puis une *classification possible des systèmes d'assemblage* ainsi qu'un ensemble de *caractéristiques spécifiques de ces systèmes* seront exposés.

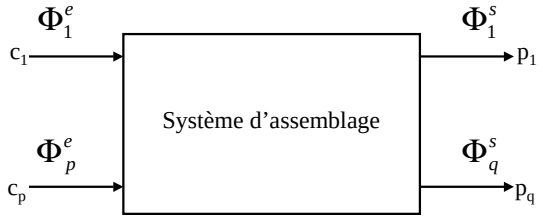


FIG. 1.1 – Système d'assemblage

1.1.1 Définition des systèmes d'assemblage

D'après J.M. HENRIOUD [30] :

DÉFINITION 1.1 (SYSTÈME D'ASSEMBLAGE)
Un système d'assemblage *est un système comportant en entrée p flux* $\Phi_i^e (i = 1, \ldots, p)$*, chacun portant sur des objets* c_i *tous identiques ; et en sortie q flux* $\Phi_j^s (j = 1, \ldots, q)$*, chacun portant sur des objets* P_j *également tous identiques.*

Comme le montre la figure 1.1, les objets c_i composant les p flux entrants sont appelés *composants élémentaires*, et les éléments P_j composant les q flux sortants sont nommés *produits finis*. Ces appellations sont relatives au système considéré. Les produits finis d'un système peuvent, par exemple, être des composants élémentaires d'un système aval. Si le nombre q de flux de sortie est égal à 1 alors c'est un *système mono-produit*, si q est supérieur à 1 alors nous parlons ici de *système multi-produits*, en assemblage, dans ce cas nous parlons aussi de *familles de produits*.

1.1.2 Classification des systèmes d'assemblage

Un système d'assemblage est une suite de postes séparés par des convoyeurs (automatiques, manuels, etc.) A chaque poste est affecté un ensemble d'opérations et de ressources. D'après G. BOOTHROYD [9], les systèmes d'assemblage peuvent être répartis en trois grandes classes :

- *Système d'assemblage manuel :*
  Dans un système manuel, toutes les opérations d'assemblage sont effectuées par des opérateurs humains, plus ou moins assistés par des ressources. Les outillages sont simples et peu coûteux, de plus les opérateurs peuvent réagir très vite aux aléas. Ce sont des systèmes très réactifs dédiés aux multi-produits.
- *Système d'assemblage automatisé spécialisé :*
  Les systèmes d'assemblage automatisés spécialisés sont des systèmes d'assemblage

de grande série d'un produit spécifique. Il est quasiment impossible de changer de produit en cours de production, pour cela, il faut interrompre le système pendant des périodes relativement longues et avec des coûts d'arrêt de production et des coûts de restructuration importants. Ce sont des systèmes très rigides et avec peu de personnel. Ils sont dédiés au mono-produit et généralement à la production de masse.

– *Système d'assemblage flexible :*
  Les systèmes flexibles sont des systèmes très coûteux, car ils nécessitent des outillages spécifiques (robots, chaînes de distribution, contrôles, ...), mais ils réagissent très rapidement et efficacement aux aléas et aux changements de produits. Ce sont des systèmes dédiés aux multi-produits. Mais pour cela il est nécessaire de prévoir des changements d'*outils* faciles et rapides.

1.1.3 Caractéristiques spécifiques des systèmes d'assemblage

Une classification des caractéristiques spécifiques des systèmes d'assemblage fait ressortir quelques inconvénients :

– *Multiplicité des variables :* les systèmes sont multi-variables car ils ont au moins deux entrées et possèdent plusieurs flux de matière convergents. En assemblage, se pose le problème de la synchronisation de ces flux.
– *Multiplicité des processus d'assemblage :* Pour l'assemblage d'un produit, il existe, en général, une multitude de possibilités d'assemblage. Chacune de ces possibilités a un flux différent des autres, ce qui pose le problème de la sélection du processus d'assemblage optimal lors de la conception du système d'assemblage.
– *Temps opératoire d'assemblage :* Comme les temps opératoires, en assemblage, sont du même ordre que les temps de convoyage entre postes, l'implantation des systèmes d'assemblage et le choix des systèmes de transfert sont très importants.
– *Hétérogénéité des équipements :* Il existe des centres d'usinage, mais en assemblage ces centres universels n'existent pas. Il faut alors créer des machines « spéciales » ou modifier et combiner plusieurs machines existantes, car les produits sont très variés dans leur forme, leur masse, leur matière, etc.
– *Présence fréquente d'aléas :* Le grand nombre de flux dans un système d'assemblage et le grand nombre d'équipements impliquent un nombre relativement important de pannes ou d'arrêts de la production.
– *Multi-produits :* Pour un assemblage multi-produits, il faut que la famille des produits assemblés ait un minimum de propriétés en commun.

1.2 Conception de systèmes d'assemblage

Chaque système d'assemblage est composé de deux grandes catégories de ressources : les *opérateurs* et les *équipements périphériques*. Les opérateurs réalisent des opérations constitutives (chargement du composant de base, déchargement du produit fini, transport, etc.) Ils sont soit *universels*, soit *spécialisés*. Les opérateurs dits universels sont capables de s'adapter, et de réaliser un ensemble d'opérations selon les outils dont ils disposent, se sont des humains, des robots... Les opérateurs spéciaux sont des machines très spécifiques ne pouvant faire qu'un ou deux types d'opérations (presses, machine à souder...). Quant aux équipements périphériques, se sont des systèmes qui n'interviennent pas dans la fabrication directe du produit fini, comme par exemple, les bols vibrants, les systèmes de transferts à bande ou vibrants, les stocks de pièces ou d'outillage, etc.

En assemblage, un système se décompose en îlots d'assemblage, qui se décomposent en cellules d'assemblage, qui elles-mêmes se décomposent en postes d'assemblage. Généralement un poste d'assemblage traite un seul produit à la fois, le produit passe de poste en poste par des systèmes de transfert. Un ensemble de postes et de systèmes de transfert forme une cellule d'assemblage. Cette cellule est relativement peu autonome dans le temps, avec un stock de quelques pièces, ce qui permet tout de même de gérer un certain nombre d'imprévus. L'ensemble des cellules travaillant sur le même produit s'appelle un îlot. Il traite un produit depuis les composants élémentaires jusqu'au produit fini.

Les systèmes d'assemblage dépendent principalement des :

- *critères d'optimisation* : minimisation du nombre de postes, minimisation du temps de cycle[1], minimisation des coûts d'investissement et de fonctionnement, équilibrage du temps de travail des opérateurs humains...
- *contraintes physiques* du système :
  - *contraintes d'antériorité entre les opérations* : elles déterminent les processus d'assemblage admissibles ;
  - *contraintes sur les équipements* : elles déterminent le nombre et le type d'outils disponibles pour chaque équipement ;
  - *contraintes spatiales* : chaque poste, chaque opérateur, chaque système de transfert occupe un espace minimal, la somme de ces espaces ne doit pas dépasser l'espace total utilisable ;

[1] Durée normale séparant les apparitions successives de deux produits intermédiaires à la sortie du poste considéré.

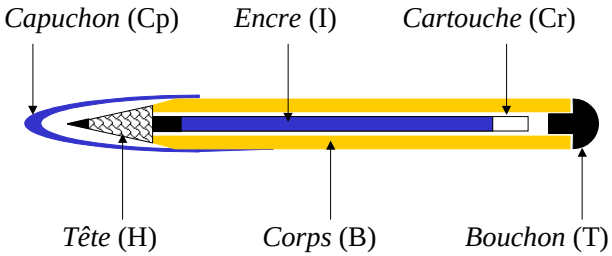


FIG. 1.2 – Stylo-bille

– *contraintes temporelles* : un certain volume de production est attendu pour le système demandé, cela se traduit par une limitation supérieure du temps de cycle et donc une réduction du temps de traitement[2], ce qui influe sur les équipements, changements d'outils, nombre de machines, espace par exemple.

Il est évident que les contraintes physiques sont relativement liées, et dépendent aussi des critères d'optimisation. Par exemple si l'espace utilisé veut être réduit, il faut des équipements plus « concentrés », plus modulables, et donc plus chers.

Lors de la conception, nous essayons de déterminer, à l'aide d'une méthode de conception, un certain nombre de paramètres comme :

– *le nombre de postes nécessaires ;*
– *le type d'opérateur à affecter à chaque poste ;*
– *les outils de chaque opérateur ;*
– *les opérations affectées à chaque poste.*

La conception de systèmes d'assemblage est constituée des étapes suivantes : la *modélisation du produit*, la *modélisation des processus d'assemblage*, *modélisation des ressources d'assemblage* et *affectation des opérations aux différents postes*. Un rappel, consacré à ces modélisations et affectations est proposé dans les sections suivantes et sera illustré à l'aide de l'exemple simple de la figure 1.2.

1.3 Modélisation des produits

A. DELCHAMBRE [16] a classifié les différents modèles de représentation des produits en deux grands groupes :

[2]Durée pendant laquelle une instance du produit subit le traitement caractéristique du poste considéré.

- *Modélisation géométrique :* elle est basée sur une description géométrique et spatiale des composants élémentaires du produit, cette modélisation est principalement liée à des logiciels de CAO[3] ;
- *Modélisation relationnelle :* seules les relations entre les composants élémentaires du produit sont décrites.

Dans ces travaux, nous ne parlerons que de modélisation relationnelle, qui est très utilisé par A. Bourjault, T.L. De Fazio, L.S. Homen de Mello, J.M. Henrioud, A. Delchambre, K. Chen, P. De Lit. La modélisation géométrique est plutôt utilisée par des chercheurs en *méthodologie d'usinage* comme par exemple K. Mawussi [38], [39].

Nous rappelons ici les définitions de A. Bourjault [10] en assemblage.

Définition 1.2 (Composant élémentaire)
Un composant élémentaire *est un objet entrant dans un système d'assemblage (voir figure 1.1). Il peut être un objet élémentaire (résultant d'un processus de fabrication —usinage, moulage...—) ou être un objet complexe produit par un autre système d'assemblage.*

La figure 1.2 montre l'ensemble des composants élémentaires du stylo-bille, qui sont le capuchon (Cp), la tête (H), l'encre (I), le corps (B), la cartouche (Cr) et le bouchon (T).

Définition 1.3 (Produit fini)
Un produit fini *est un objet de sortie d'un système d'assemblage.*

Définition 1.4 (Constituant)
Un constituant est tout objet intermédiaire apparaissant lors d'un processus d'assemblage. Cette définition inclut les composants élémentaires, le produit fini et tous les objets intermédiaires.

Avec la figure 1.4, le produit est composé de trois constituants, par exemple le sous-ensemble *« tête, cartouche, encre »* en est un.

Définition 1.5 (Liaison)
Il existe une liaison *et une seule entre deux composants élémentaires c_i et c_j d'un produit donné si et seulement si il existe au moins une liaison mécanique entre ces deux composants.*

La figure 1.3 représente les liaisons entre les composants élémentaires du stylo-bille sous la forme d'un graphe, appelé *graphe des liaisons géométriques.*

[3] Conception Assistée par Ordinateur

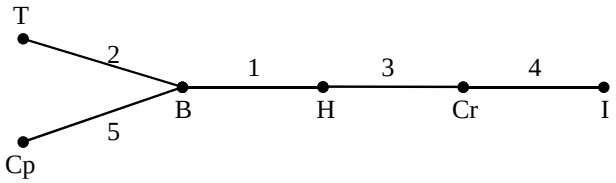


FIG. 1.3 – Graphe des liaisons géométriques du stylo-bille

DÉFINITION 1.6 (GRAPHE DES LIAISONS GÉOMÉTRIQUES)
Le graphe des liaisons géométriques *d'un produit P est un graphe simple et non orienté noté $G = [C, L]$ où :*

- *C est l'ensemble des nœuds du graphe qui représentent l'ensemble des composants élémentaires du produit P (voir figure 1.3, où $C =\{$ T, Cp, B, H, Cr, I $\}$).*
- *L est l'ensemble des arêtes du graphe qui caractérisent l'ensemble des liaisons géométriques du produit P (voir figure 1.3, où $L =\{$ 1=(B, H), 2=(B, T), 3=(H, Cr), 4=(Cr, I), 5=(B, Cp) $\}$).*

Il y a dans la figure 1.3, une représentation du graphe $G = [\{ T, Cp, B, H, Cr, I\}, \{ 1, 2, 3, 4, 5 \}]$

DÉFINITION 1.7 (ACTION)
Une action *entre deux constituants est l'établissement d'une liaison et une seule.*

A l'aide des composants élémentaires *« corps »* et *« bouchon »* de la figure 1.2, il est possible de dire que l'action de solidarisation du *« bouchon »* sur le *« corps »* est l'insertion du *« bouchon »* dans le *« corps »*. Cette action réalise alors la liaison 2 du graphe des liaisons de la figure 1.3.

DÉFINITION 1.8 (SOUS-ASSEMBLAGE)
Un sous-ensemble (X, α) (où x est un ensemble de constituants et α est l'ensemble des liaisons associées) est un sous-assemblage *si et seulement si :*

- *il peut être assemblé, c'est-à-dire que l'établissement de toutes les liaisons fonctionnelles de α peut être effectué et ce, indépendamment des liaisons de $L - \alpha$ (voir le sous-assemblage de la figure 1.4 « tête, cartouche, encre » où $X = \{T, Cp, B, H, Cr, I\}$ et $\alpha = \{3, 4\}$).*
- *il est possible, à partir de (X, α) de réaliser le produit fini ; c'est-à-dire que l'ensemble $L - \alpha$ des liaisons restantes pourra être établi lorsque les liaisons de α seront réalisées (voir figures 1.3 et 1.4 où $X = \{T, Cp, B, H, Cr, I\}$ et $L - \alpha = \{1, 2, 5\}$).*

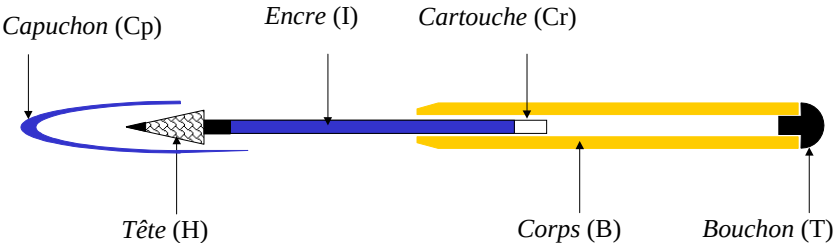


FIG. 1.4 – Assemblage du stylo-bille avec sous-assemblages

La figure 1.4 fait apparaître deux sous-assemblages : le sous-assemblage *« tête, cartouche, encre »* et le sous-assemblage *« corps, bouchon »*).

DÉFINITION 1.9 (SOUS-ASSEMBLAGES INDÉPENDANTS)
Deux sous-assemblages *sont dits* indépendants *l'un de l'autre si :*

- *ils n'ont ni composants ni liaisons en commun (voir figure 1.4) ;*
- *l'état obtenu après la réalisation de ces deux sous-assemblages est un état à partir duquel il est possible d'atteindre l'état final (voir figure 1.4 : les deux sous-assemblages* « tête, cartouche, encre » *et* « corps, bouchon » *permettent d'obtenir le produit fini).*

DÉFINITION 1.10 (ÉTAT DU PRODUIT INTERMÉDIAIRE)
Nous appelons état du produit intermédiaire *l'ensemble des liaisons déjà établies à une étape quelconque du processus d'assemblage.*

Un état intermédiaire est considéré comme *admissible* s'il peut être obtenu depuis un pré-produit et s'il permet d'obtenir un produit fini. Le pré-produit est défini par l'état où aucune liaison fonctionnelle n'est établie, il est noté $(\bar{1}\ldots,\bar{n})$. Le produit fini est noté $(1\ldots,n)$, où toutes les liaisons sont établies.

Prenons notre exemple de la figure 1.2, $(\bar{1},\bar{2},\bar{3},\bar{4},\bar{5})$ est le pré-produit, et l'état intermédiaire $(1,\bar{2},3,4,\bar{5})$ représente le sous-assemblage *« Tête, Cartouche, Encre, Corps »*.

Nous donnons ci-après le plus complet des modèles relationnels existants qui est la définition du modèle du produit proposée par J.M. HENRIOUD. Il est appelé *modèle opératoire* car il prend en compte l'ensemble de toutes les opérations qui peuvent être effectuées lors du processus d'assemblage.

DÉFINITION 1.11 (MODÈLE DU PRODUIT FINI À L'AIDE DES CARACTÈRES)
Soit un produit P modélisé par un 5-uplet $< C, L, \Sigma, \Delta, h >$. L'assemblage de ce produit est réalisé par l'établissement successif de l'ensemble de caractères $L \cup \Sigma \cup \Delta$. A une étape quelconque de ce processus, l'ensemble de caractères déjà établi constitue un état intermédiaire *du produit P.*

- *C est l'ensemble des composants élémentaires,*
- *L est l'ensemble des liaisons,*
- *Σ est l'ensemble des solidarisations (soudure, collage...),*
- *Δ est l'ensemble des caractères complémentaires (contrôle, marquage...),*
- *h est l'ensemble des conditions nécessaires d'établissement de tous les caractères non géométriques : solidarisations et caractères complémentaires.*

Avec l'exemple de la figure 1.2 : nous avons $C = \{T, Cp, B, H, Cr, I\}$, $L = \{1, 2, 3, 4, 5\}$, $\Sigma = \emptyset$, $\Delta = \emptyset$, $h = \emptyset$.

1.4 Modélisation des processus d'assemblage

Le processus qui transforme un ensemble de composants élémentaires en un produit fini est appelé *processus d'assemblage*.

Une modélisation[4] des processus d'assemblage est un moyen de modéliser et de mettre en évidence un ensemble de conditions et d'enchaînement des opérations ou des actions d'assemblage.

Par définition, une représentation est dite *valide* si et seulement si elle vérifie la condition : *Tous les processus représentés sont admissibles* : ils permettent tous, à partir d'un pré-produit, d'obtenir, après un nombre fini d'étapes, le produit fini.

Une représentation est dite *exhaustive* si et seulement si elle vérifie la condition : *Tous les processus d'assemblage admissibles sont représentés.*

Nous noterons toutefois que, contrairement à la conception de systèmes d'assemblage, le pilotage de systèmes d'assemblage ne nécessite pas l'exhaustivité d'une représentation. Cette non nécessité est due généralement à une utilisation restreinte des possibilités des équipements ou à l'incapacité de mettre en œuvre certains processus admissibles pour des raisons financières ou autres.

Mais en ce qui concerne la validité, elle doit être présente dans tous les cas, ce qui nous amène à comparer les différentes représentations existantes selon qu'elles sont utilisées en conception ou en pilotage de systèmes d'assemblage :

- *dans la conception des systèmes d'assemblage* : Selon C. Perrard [45] pour résoudre les problèmes d'exhaustivité, il est nécessaire d'avoir des connaissances sur

[4] Nous parlerons aussi de représentation.

les opérations, sur les processus d'assemblage, sur les paramètres de production, sur les ressources disponibles. La représentation doit pouvoir :

1. Expliciter les opérations d'assemblage ;
2. Exprimer les divergences des différents processus.

– *dans le pilotage de système d'assemblage* : Le pilotage peut être traduit par un respect des contraintes de production (ordonnancement, temps...), ou par un respect du rythme de production, ou par des réponses rapides (réactivité). Et tout cela traduit une utilisation efficace des ressources. Le pilotage est une gestion *dynamique* des décisions à prendre pour profiter au maximum de la flexibilité du système. Cette méthode demande à la représentation d'être relativement claire du point de vue de l'identification des opérations déjà effectuées et de celles qui restent à traiter à tout moment.

Par ce souci du respect des temps de cycle, les diverses *dépendances temporelles* entre les opérations d'assemblage ont été étudiées. Nous appelons *dépendances temporelles* entre deux opérations, le fait que la réalisation de l'une conditionne la réalisation de l'autre, ce qui se traduit par une contrainte de précédence[5]. Nous appelons *indépendance temporelle* entre deux opérations, le fait que l'ordre d'exécution soit libre et qu'elles puissent être faites en parallèle. Selon, C. PERRARD [45], il est toujours possible de traduire les processus d'assemblage par un ensemble d'opérations d'assemblage sous un ordre partiel en mettant en évidence la *dépendance* ou l'*indépendance temporelle* des opérations d'assemblage. Lors du pilotage, la mise en évidence des dépendances est rapide et facile. Contrairement à ce qui se passe en conception, elle dépend uniquement de la *lisibilité* de la représentation. Cette *lisibilité* est l'écriture claire et pratique, pour un expert humain, d'une représentation, elle est inversement proportionnelle à la *compacité*. Celle-ci traduit la capacité de la représentation à exprimer le maximum d'informations avec un minimum d'éléments.

D'après cette précision, il est possible de dire qu'une « bonne » représentation est une représentation qui est *valide*, *exhaustive*, *utile*, *compacte* et exprimant les *dépendances temporelles* tout en étant *lisible*.

L'ensemble des représentations des processus d'assemblage peut se diviser en deux grands groupes, selon que nous considérons le processus comme une suite d'établisse-

[5]Une contrainte de précédence entre deux opérations a et b est exprimée par la précédence de l'opération a sur l'opération b dans les processus d'assemblage, noté $a \rightarrow b$.

ments de liaisons (*approche liaisons*) ou comme une suite d'adjonctions de composants (*approche composants*).

1.4.1 Approche liaisons

Selon la définition 1.7 p.9, une action est la réalisation d'une liaison et une seule. Il existe deux grandes représentations basées sur les actions : les *séquences d'assemblage* et les *graphes OU*.

1.4.1.1 Séquence d'assemblage

Les séquences d'assemblage ont été introduites par A. Bourjault [10], qui en donne la définition suivante :

Définition 1.12 (Séquence d'assemblage)
Une séquence d'assemblage *est une suite ordonnée de $m-1$ commandes permettant d'établir le produit fini P, m étant le nombre de composants élémentaires du produit.*

Définition 1.13 (Commande)
Une commande *est une action, ou ensemble d'actions réalisées simultanément, permettant de passer d'un état intermédiaire admissible à un autre, sans qu'il y ait d'état intermédiaire entre ces deux états.*

Il est très important de noter la différence entre une commande et une action. Une action réalise une liaison entre deux constituants, tandis qu'une commande réalise toutes les liaisons du dernier constituant apporté à l'assemblage avec les constituants déjà présents dans le *sous-assemblage* existant. La figure 1.5 montre l'ensemble des séquences d'assemblage du stylo-bille.

1.4.1.2 Graphes OU

T.L. De Fazio et D.E. Whitney [14] ont introduit les graphes OU en 1987. Comme le montre la figure 1.6, un *graphe OU* est un graphe orienté où :

- chaque nœud est un état intermédiaire ;
- chaque arc représente une commande (voir définition 1.13 page 13).

Les *graphes OU* sont un autre moyen de représenter les séquences d'assemblage. Le nœud initial représente le pré-produit (l'ensemble des composants élémentaires à l'entrée du

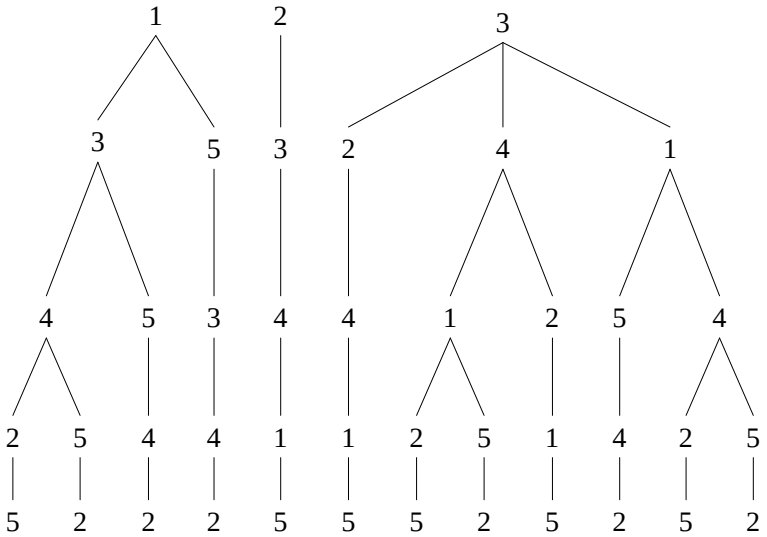


FIG. 1.5 – Séquences d'assemblage du stylo-bille

système d'assemblage), le nœud final représente le produit fini. Chaque chemin du nœud initial au nœud final représente une séquence d'assemblage.

1.4.1.3 Évaluation

Malgré certains avantages comme la validité et l'exhaustivité sous forme de suites séquentielles ordonnées de commandes, les *séquences d'assemblage* ont un grand nombre d'inconvénients. Elles ne sont pas compactes (voir figure 1.5) donc difficiles d'utilisation, de plus il est quasiment impossible d'intégrer des informations ultérieurement. Les figures 1.5 et 1.6 montrent la non lisibilité des processus d'assemblage, l'absence des contraintes de précédences, la difficulté d'établir les états intermédiaires. Et contrairement à l'homme, les séquences d'assemblage et les graphes OU traitent les commandes et non les opérations (voir définition 1.14 ci-après). Il est difficile d'envisager l'application de ces deux représentations tant en conception qu'en pilotage des systèmes d'assemblage.

1.4.2 Approche composants

Les approches composants sont généralement basées sur la représentation des opérations d'assemblage.

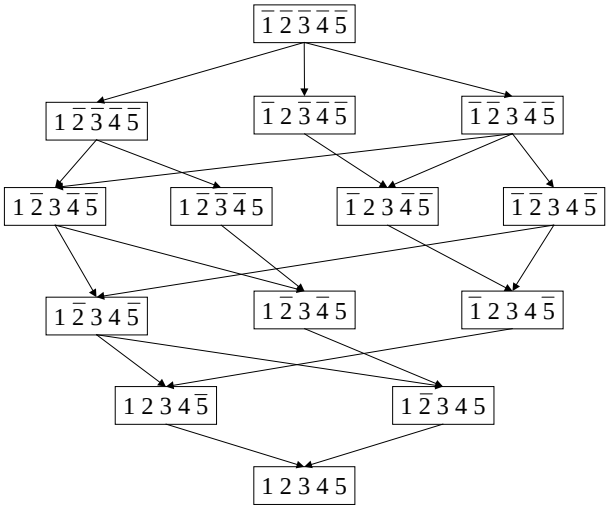


FIG. 1.6 – graphes OU du stylo-bille

DÉFINITION 1.14 (OPÉRATION D'ASSEMBLAGE)
Une opération d'assemblage est la production d'un constituant soit par l'union de deux constituants, soit par l'adjonction d'un caractère non géométrique à un constituant.

1.4.2.1 Arbres et graphes d'assemblage

J.M. HENRIOUD [30] utilise les arbres d'assemblage, qui sont des arborescences dont :
- la racine est le produit fini P ;
- les nœuds non terminaux sont des sous-assemblages ;
- les feuilles sont des composants élémentaires ou des caractères non géométriques.

Toute opération d'assemblage géométrique consiste en la réunion de deux objets. L'un, en général, est immobile pendant l'opération d'assemblage, il est appelé *constituant primaire* tandis que l'autre, appelé *constituant secondaire*, subit un déplacement. De plus, lorsque le constituant primaire est un composant élémentaire, il est appelé *composant de base*.

Une opération d'assemblage peut alors être représentée par un couple (S, e) où :
- S est le constituant primaire ;
- e est soit un constituant secondaire, soit un caractère non géométrique.

Suite aux travaux de J.M. HENRIOUD, V. MÎNZU [41] a introduit les graphes d'assemblage dont il donne la définition suivante :

DÉFINITION 1.15 (GRAPHE D'ASSEMBLAGE)
Un graphe d'assemblage est une anti-arborescence dont :

- *la racine est l'opération de déchargement du produit fini, notée u ;*
- *les feuilles sont les opérations de chargement du composant de base B_i des différents sous-assemblages, notées aussi B_i ;*
- *les nœuds intermédiaires sont les opérations d'assemblage (S, α_i), notées α_i.*

DÉFINITION 1.16 (OPÉRATION)
Une opération est la réalisation par un opérateur d'une opération d'assemblage, qu'elle soit constitutive ou logistique. Lors de la conception de système, seuls trois types d'opérations ont un intérêt pour nous :

- *les opérations constitutives : réalisation d'opérations constitutives (adjonction d'un constituant secondaire ou réalisation d'un caractère non géométrique) ;*
- *les opérations de chargement : réalisation de l'opération de chargement du composant de base ;*
- *les opérations de déchargement : réalisation de l'opération de déchargement du produit fini.*

Dans un graphe d'assemblage comme dans toute représentation nécessitant la détermination d'un constituant primaire, comme les LASTD[6], les graphes de précédence ou les P-Q-R arbres (voir paragraphes suivants), le constituant primaire est connu à chaque étape. Toute opération géométrique est représentée par le constituant secondaire, toute opération non géométrique est représentée par le caractère concerné, toute opération de chargement est représentée par le composant de base manipulé, l'unique opération de déchargement est symbolisée par u.

D'après les définitions de J.M. HENRIOUD et de V. MÎNZU, nous pouvons dire que les graphes d'assemblage et les arbres d'assemblage sont des représentations relativement proches. Dans les arbres d'assemblage, les constituants primaires ne sont pas définis, les nœuds représentent des sous-ensembles, tandis que dans les graphes d'assemblage, les nœuds représentent des opérations dont le constituant secondaire (ou le caractère non géométrique) est explicitement représenté par l'étiquette du nœud correspondant et le constituant primaire implicitement représenté par l'union de tous les nœuds prédécesseurs. Les graphes d'assemblage ont l'avantage sur les arbres d'assemblage de mettre plus facilement en évidence les sous-assemblages.

[6] Layer Assembly State Transition Diagram *ou* Graphes d'état des transitions d'assemblage

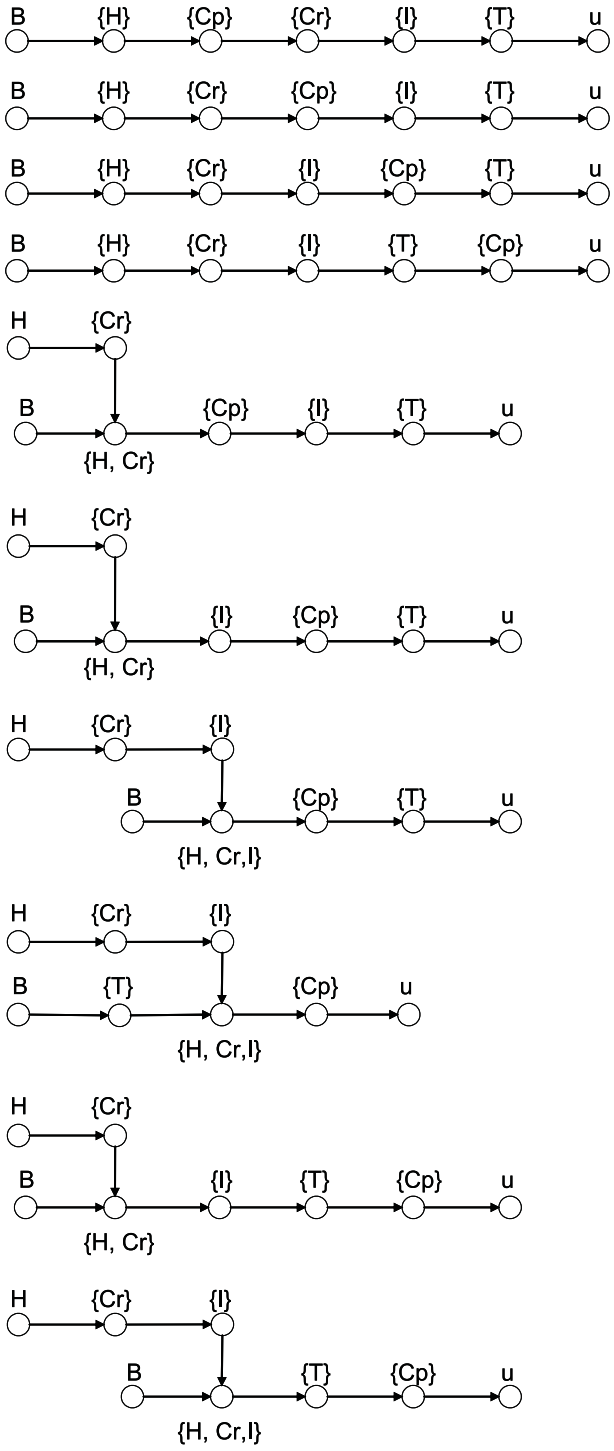


FIG. 1.7 – Graphes d'assemblage du stylo-bille

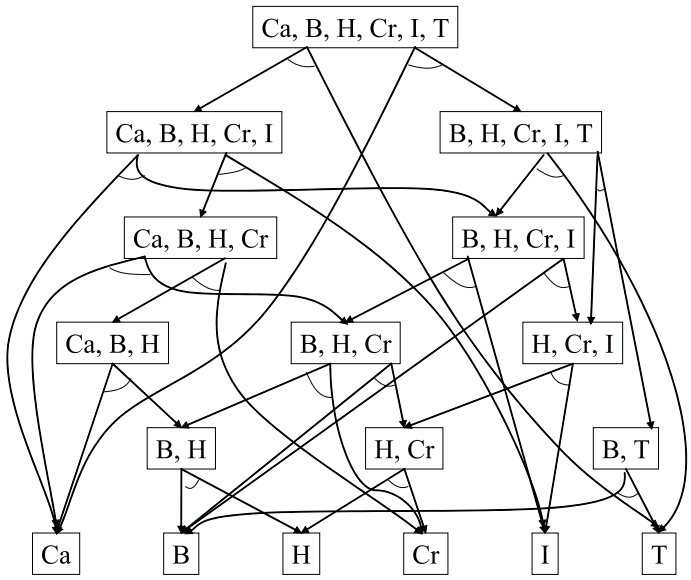


FIG. 1.8 – Graphe ET/OU du stylo-bille

1.4.2.2 Graphes ET/OU

En 1989 L.S. HOMEM DE MELLO et A.C. SANDERSON [35] ont introduit le graphe ET/OU, qui est une représentation collective des arbres d'assemblage où :

- chaque nœud représente d'une façon unique un sous-assemblage du produit fini ;
- chaque opération s'exprime sous la forme d'un hyperarc : le nœud initial représente le constituant résultant, et les successeurs représentent les constituants à assembler.

Les graphes ET/OU possèdent les avantages des arbres d'assemblage. Mais la lisibilité de cette représentation collective est réduite (voir figure 1.8). De plus son utilisation n'est pas évidente ni en conception ni en pilotage de système.

1.4.2.3 Réseaux de Petri

Un réseau de Petri est défini par un 4-uplet P, T, Pré, Post avec :

- P : l'ensemble des places ;
- T : l'ensemble des transitions ;
- Pré : l'application d'incidence avant entre les transitions et les places : $P \times T \rightarrow 0,1$;
- Post : l'application d'incidence arrière entre les places et les transitions : $T \times P \rightarrow 0,1$.

Il est possible de transformer les arbres d'assemblage en réseaux de Petri, en attribuant les constituants aux places et les opérations aux transitions.

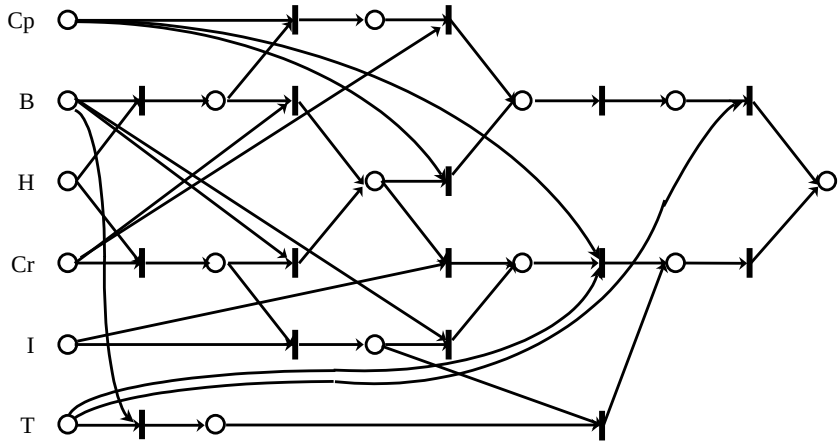


FIG. 1.9 – Réseau de Petri du stylo-bille

Les réseaux de Petri sont les représentations les plus adaptées au pilotage de système, cependant nous ne connaissons pas actuellement de méthode pertinente d'équilibrage de charge pour la conception de système basée sur cette représentation. La figure 1.9 est un réseau de Petri représentant l'assemblage du stylo-bille de la figure 1.2.

1.4.2.4 Graphes de précédence

D'après T.O. PRENTING [46] :

DÉFINITION 1.17 (GRAPHE DE PRÉCÉDENCE)
Un graphe de précédence est un graphe simple, orienté, connexe, sans boucle et sans circuit dont :

- *les nœuds sont des tâches ;*
- *les arcs décrivent les contraintes d'antériorité entre les tâches.*

DÉFINITION 1.18 (TÂCHE)
Une tâche est la réalisation par un opérateur d'une opération d'assemblage, qu'elle soit constitutive ou logistique. Lors de la conception de système, seuls trois types de tâches ont un intérêt pour nous :

- *les tâches constitutives : réalisation d'opérations constitutives (adjonction d'un constituant secondaire ou réalisation d'un caractère non géométrique) ;*
- *la tâche de chargement : réalisation de l'opération de chargement du composant de base ;*
- *la tâche de déchargement : réalisation de l'opération de déchargement du produit fini.*

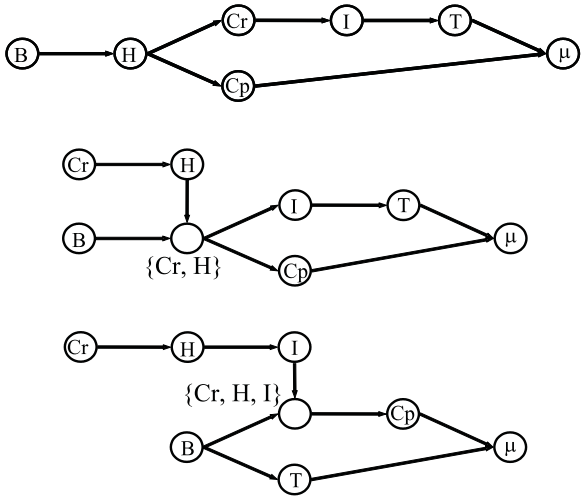


FIG. 1.10 – Graphes de précédence du stylo-bille

Il est possible de dire alors que l'ensemble des tâches d'assemblage associées à un graphe de précédence représente l'ensemble des tâches d'assemblage augmenté de la tâche de déchargement et de celle de chargement.

REMARQUE 1.1
Nous noterons les tâches d'assemblage d'un constituant par le nom du dit constituant (voir figure 1.10).

La figure 1.10 montre l'ensemble des graphes de précédence du stylo-bille.

1.4.2.5 Assembly State Transition Diagram (ASTD)

C.J.M. HEEMSKERK et N. BONESCHANSCHER[5], [7], [29], ont introduit les ASTD[7] ainsi qu'une extension, les LASTD[8].

DÉFINITION 1.19 (ASTD)
Un ASTD est un graphe direct composé de nœuds et d'arcs, et ne possédant pas de cycle.

- *chaque nœud représente un état du produit intermédiaire, où un constituant X est noté X s'il est déjà assemblé et $\bar{X}$ dans le cas contraire ;*
- *chaque arc, reliant deux nœuds, est orienté et représente la commande nécessaire pour passer de l'état du nœud origine à l'état du nœud destination.*

[7] Assembly State Transition Diagram *ou* Graphes d'état des transitions d'assemblage de simple niveau
[8] Layer Assembly State Transition Diagram *ou* Graphes d'état des transitions d'assemblage

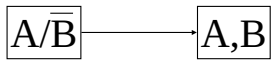


FIG. 1.11 – Précédence dans un ASTD.

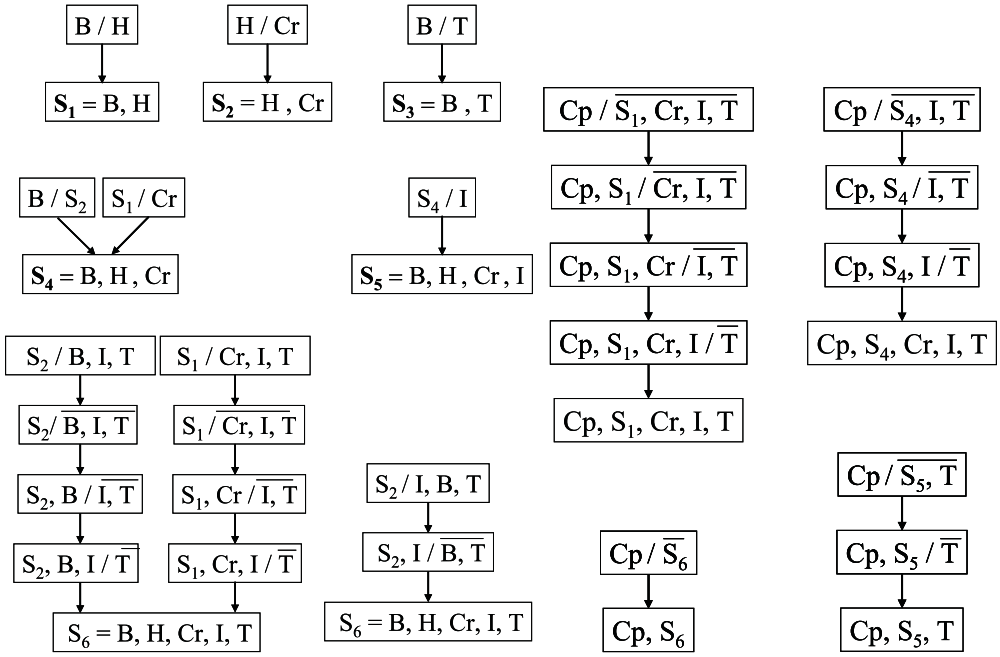


FIG. 1.12 – LASTD du stylo-bille

L'ensemble des composants assemblés est séparé de l'ensemble de ceux qui ne le sont pas encore par le symbole «/».

Comme le montre la figure 1.11, dans le nœud initial, le constituant A est déjà intégré, mais pas le constituant B, tandis que dans le nœud final, les deux constituants sont présents.

Nous appelons LASTD l'ensemble des ASTD d'un même produit. Un LASTD est un graphe OU modifié de la manière suivante :

- le constituant primaire de chaque sous-graphe est déterminé ;
- à chaque état intermédiaire, seuls les constituants du produit en cours interviennent (voir figure 1.12) ;
- chaque sous-assemblage est un produit fini dans un ASTD et un constituant dans un autre.

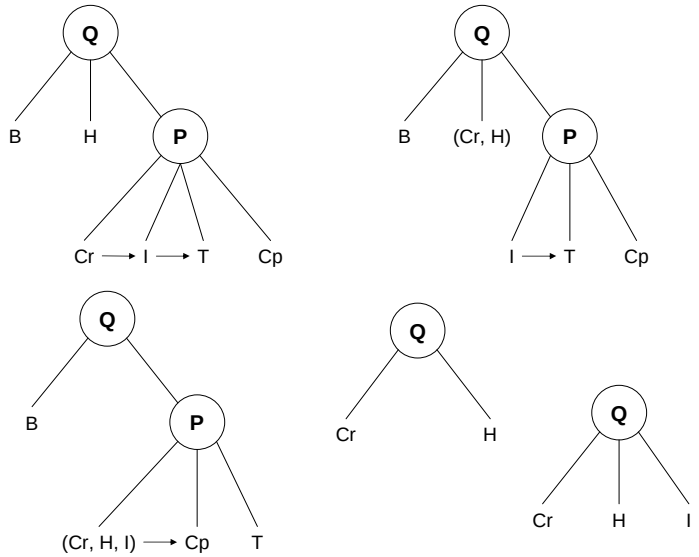


FIG. 1.13 – P-Q-R arbres du stylo-bille

1.4.2.6 P-Q-R arbres

Dans les années 1975, K.S. BOOTH [8], puis P. BAPTISTE et ses collègues [2] en 1991, ont introduit les P-Q-R arbres et en ont donné la définition suivante :

DÉFINITION 1.20 (P-Q-R ARBRES)
Les P-Q-R arbres sont des arbres où :

- *les nœuds non terminaux sont l'un des trois opérateurs :*
  - *P décrit un ordre partiel entre les éléments de l'ensemble auquel il s'applique ;*
  - *Q définit un ordre total entre les éléments de l'ensemble auquel il s'applique ;*
  - *R représente toutes les permutations sur l'ensemble auquel il s'applique.*
- *les feuilles sont les composants élémentaires.*

Un ensemble de X séquences d'assemblage peut être représenté par un ensemble de P-Q-R arbres, mais il ne sera peut être pas unique, comme le montre la figure 1.13 représentant les P-Q-R arbres du stylo-bille.

1.4.2.7 Évaluation

Les *arbres d'assemblage* sont une représentation compacte des séquences d'assemblage. Ils sont représentatifs de la structure du produit. Ils offrent une bonne lisibilité du

fait que ce sont les constituants qui sont représentés et non les commandes. Ces arbres sont utilisés en conception mais très peu en pilotage, de plus leur existence pratique est relativement répandue.

La représentation par *graphes d'assemblage* n'est pas compacte et a peu d'intérêts pour les industriels, elle n'est donc pas très pertinente.

Le *graphe ET/OU* est une représentation collective des arbres d'assemblage. Il en a donc les avantages, mais comme toutes représentations collectives, sa lisibilité en est réduite. Il est relativement difficile d'utiliser cette représentation pour la conception ou la conduite de système d'assemblage de par le manque de lisibilité des processus et des opérations, et de par la non observation des contraintes de précédence. Nous pouvons dire que cette représentation est d'une qualité moyenne et très peu utilisée.

Comme vu précédemment, les *réseaux de Petri* sont une traduction des graphes ET/OU, mais cette représentation est très pertinente du point de vue du pilotage de système avec la mise en évidence des processus et des sous-assemblages. Le fait qu'ils permettent de connaître à tout moment l'état de chaque poste (disponibilité, attente, etc.) est aussi très intéressant. Cette représentation est, aujourd'hui, l'outil le plus utilisé en pilotage de systèmes.

Les *graphes de précédence* comme les *ASTD* et les *P-Q-R arbres* sont des représentations valides et exhaustives nécessitant la définition d'un composant de base. De plus, elles sont toutes les trois des représentations mettant en évidence le parallélisme des opérations, avec leurs graphes distincts, elles montrent les sous-assemblages possibles. Et bien que ces représentations soient *multi-graphes*, elles n'en restent pas moins compactes.

Les *graphes de précédence* sont très lisibles au niveau des processus comme au niveau des opérations, et ils mettent en évidence les contraintes de précédence tout en faisant ressortir la notion de parallélisme. L'utilité de cette représentation n'est plus à démontrer tant en conception qu'en pilotage. Mais il faut quand même spécifier que son utilisation est nettement supérieure en conception par son nombre d'applications pratiques.

Les *ASTD* ne mettent pas les opérations d'assemblage en évidence, mais ont une certaine lisibilité au niveau des états intermédiaires, cela pourrait être utile en pilotage de système. Leur utilisation est tout de même restreinte.

Les *P-Q-R arbres* sont une représentation valide, exhaustive, relativement compacte, ayant une certaine lisibilité au point de vue des opérations, mais quasiment nulle au niveau des processus. La mise en évidence des contraintes de précédence est inexistante

ou presque. Son utilité est très faible, tant pour la conception de système que pour le pilotage.

1.4.3 Évaluation de l'ensemble des représentations présentées

Comme nous avons pu le voir une représentation est « pertinente » si elle est valide, exhaustive, compacte, lisible tant du point de vue des processus et des opérations que des états intermédiaires. Mais elle doit surtout avoir une utilité soit en conception, soit en pilotage de système. Ce tour d'horizon, nous permet de dégager les deux représentations les plus pertinentes : les *graphes de précédence* et les *réseaux de Petri* sur l'ensemble de ces points.

Passons maintenant à l'étude des méthodes de conception des systèmes d'assemblage.

1.5 Modélisation des ressources d'assemblage

La modélisation des ressources d'assemblage est la définition du nombre de postes disponibles, du type de postes (réactifs, temps de reconfiguration...), des capacités de chaque poste et du temps de cycle. Nous voyons que ce type de modélisation représente des caractéristiques techniques du système, nous ne nous y attarderons pas.

1.6 Conclusion du chapitre

Travaillant dans une équipe de conception intégrée, nous avions comme objectif d'étudier plus particulièrement des outils pouvant apporter des éléments intéressants pour la conception de système. Cet objectif peut s'illustrer avec la figure 1.14, qui représente de manière schématique le découpage des entreprises manufacturières. Notre apport portera sur les bureaux des méthodes en assemblage, plus précisément sur la génération des représentations des processus d'assemblage. Parmi cet ensemble de représentations possibles, nous pouvons dire que la plus pertinente est la représentation par graphes de précédence. C'est un fait, les réseaux de Petri sont très utilisés en gestion de flux et en maintenance et sûreté de fonctionnement, mais les graphes de précédence sont actuellement l'outil d'aide à la conception le plus utilisé. Le graphe PERT, très utilisé en ordonnancement, est en fait lui aussi, un graphe de précédence. Cette pertinence des graphes de précédence est due à leur capacité à mettre en valeur les contraintes de précédence entre les opérations,

leur lisibilité et surtout leur utilité en conception et pilotage de système d'assemblage. Cette utilité est particulièrement décrite par :

- la mise en évidence d'un *ordre partiel* des opérations ;
- la prise en compte de plusieurs processus d'assemblage contrairement aux séquences et aux graphes d'assemblage ;
- leur compacité : généralement, quelques graphes de précédence suffisent à représenter l'ensemble des processus d'assemblage d'un produit.

Le principal défaut de cette méthode de représentation est son élaboration généralement empirique par des experts humains. Nous clôturons ce tour d'horizon des systèmes de production par la constatation qu'un manque se fait sentir au niveau de la *génération des graphes de précédence*. C'est pourquoi nous passerons les chapitres à venir sur ce point. Dans le chapitre 2 nous développerons les définitions de base nécessaires à la compréhension du chapitre 3, qui expose une *génération des graphes de précédence par évolution de graphes*. Le chapitre 4 traite d'une autre méthode de génération : la *génération des graphes de précédence par la logique booléenne*.

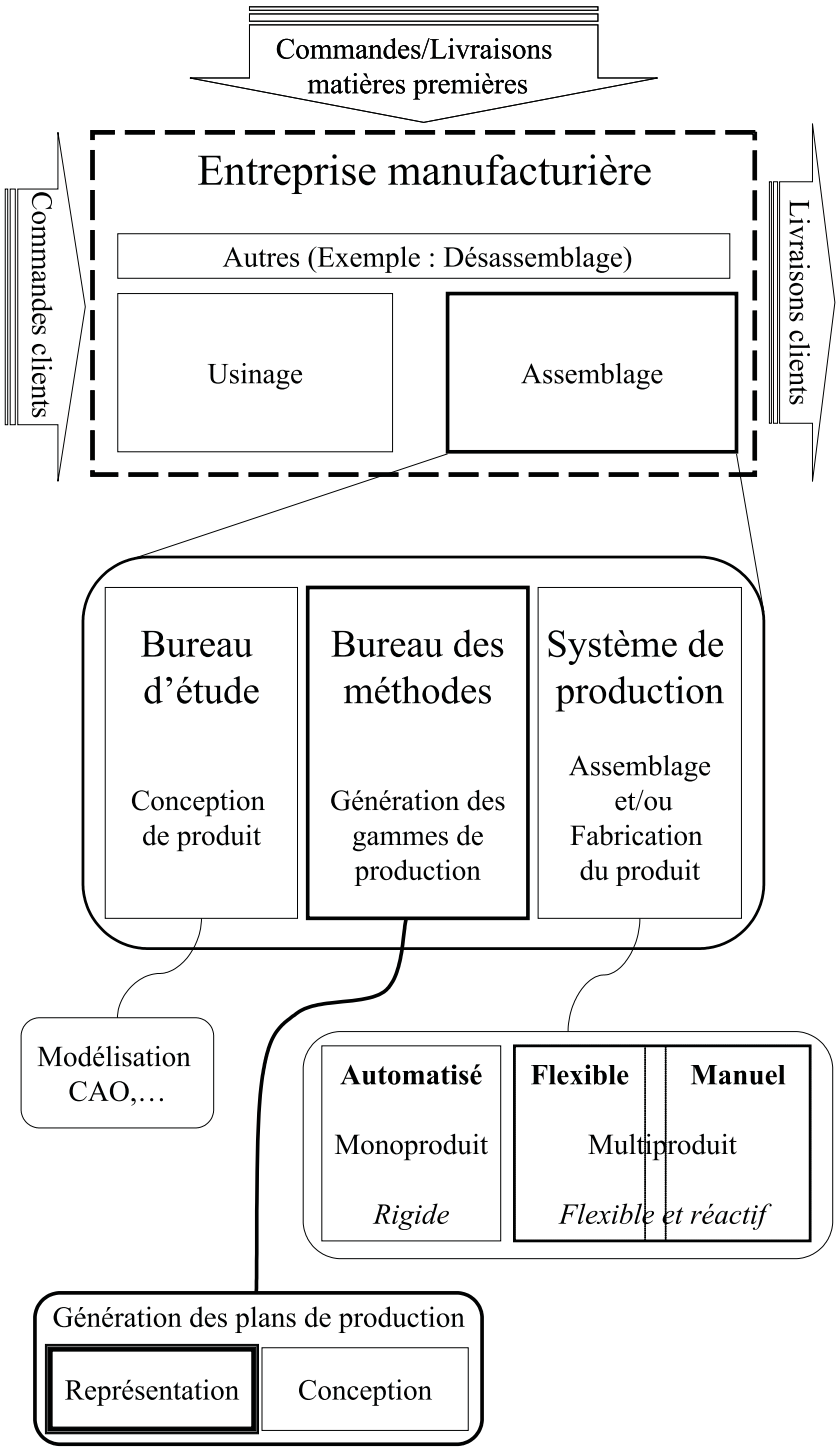


FIG. 1.14 – Position de cette étude dans le contexte industriel

Chapitre 2

Les propriétés des représentations de processus d'assemblage

Nous avons vu dans le chapitre précédent que les graphes de précédence sont largement utilisés en conception de systèmes, principalement avec les méthodes d'ALB[1]. Les besoins en terme de graphes de précédence sont importants, et malgré un grand nombre de travaux sur la génération de ces graphes, le problème ne semble toujours pas être résolu de façon satisfaisante. Ce chapitre traitera de trois grands types de représentations possibles des processus d'assemblage : les *graphes d'assemblage*, les *graphes de précédence* et les *ASTD*. Les deux premières sont liées aux opérations ou tâches et la dernière aux états du produit intermédiaire. Nous aborderons aussi une variante des graphes de précédence : les hypergraphes de précédence.

2.1 Objectif

Comme développé au cours du précédent chapitre, l'objectif de ce travail est *la génération systématique des graphes de précédence maximaux*[2] *correspondant de manière biunivoque à l'ensemble des processus d'assemblage préétablis avec LEGA.*

Après l'exposition de l'objectif et sa présentation dans la figure 2.1, la notion de processus d'assemblage sera développée ci-après.

[1] Assembly Line balancing

[2] Graphe de précédence qui génère le maximum de processus d'assemblage.

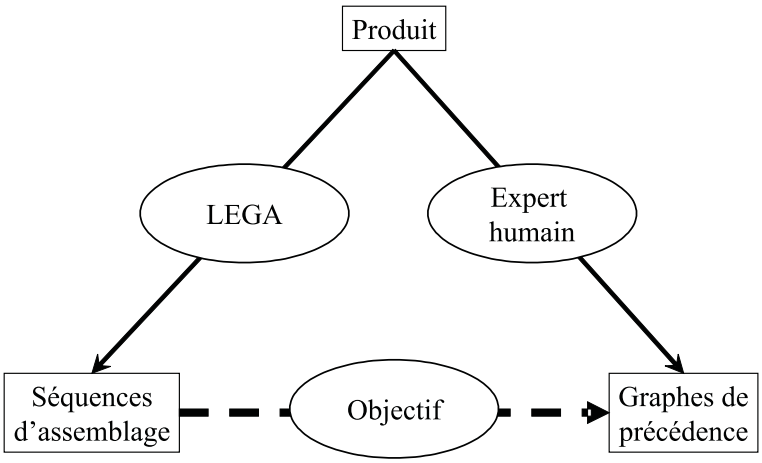


FIG. 2.1 – Objectif des travaux

2.2 Les processus d'assemblage

Les principaux points de la génération des processus d'assemblage par le logiciel LEGA[3] vont être étudiés. Cette génération est décomposée en plusieurs parties :

- la modélisation du produit,
- l'évaluation et la sélection des processus d'assemblage,
- l'implémentation de LEGA.

La modélisation du produit à assembler a été traitée dans le chapitre précédent, nous allons donc développer les deux autres points.

2.2.1 Évaluation et sélection des processus d'assemblage

Après la modélisation du produit et l'établissement des contraintes par l'expert, il est nécessaire d'évaluer les processus et de sélectionner les processus optimaux. Un processus d'assemblage est dit admissible si aucune de ses opérations d'assemblage n'est interdite par une contrainte opératoire. La détermination des processus d'assemblage admissibles est basée sur un principe de recherche de toutes les opérations réalisables, géométriques ou non, admettant pour résultat le produit fini ; à chaque opération géométrique obtenue correspond ainsi un couple de constituants et à chaque opération non géométrique correspond un couple constitué d'un caractère non géométrique et d'un constituant. Pour chaque sous-assemblage ainsi déterminé, toutes les opérations d'assemblage réalisables

[3]Logiciel d'Élaboration des Gammes d'Assemblage

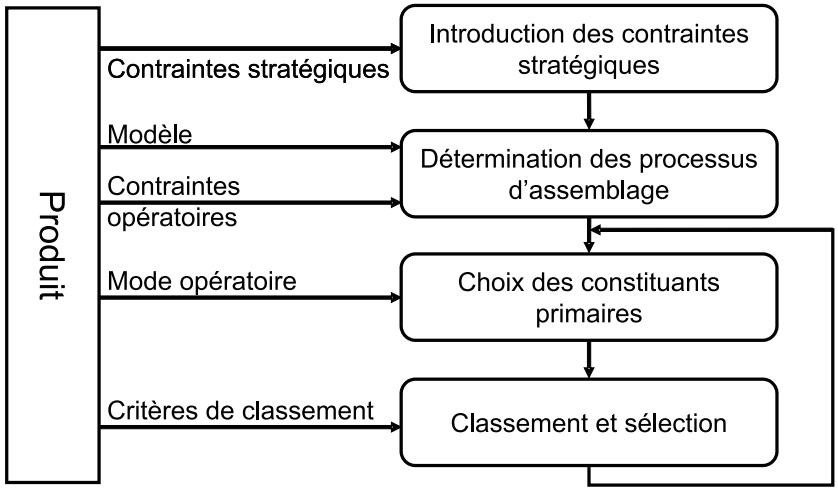


FIG. 2.2 – Méthode globale de génération et sélection des processus d'assemblage valides

l'admettant pour résultat sont recherchées. Cette démarche est répétée jusqu'à l'obtention de tous les composants élémentaires du produit considéré.

2.2.2 Implémentation

Cet algorithme a été implanté en Prolog au LAB par J.M. HENRIOUD [30], pour réaliser LEGA. Ce logiciel génère l'ensemble des processus d'assemblage admissibles pour des produits comportant au maximum une quinzaine de composants ; cette limitation étant imposée par une explosion combinatoire du nombre de processus possibles. Il existe deux grands types de sélection des processus d'assemblage, l'un utilise les contraintes stratégiques, l'autre l'évaluation. Le premier peut être introduit à deux moments différents pour des résultats similaires. Soit ces contraintes sont introduites dans le modèle avant la recherche de réalisabilité des opérations, ou à la fin de cette procédure afin de réduire le nombre de processus admissibles. Tandis que l'évaluation ne peut se faire qu'après obtention de tous les processus admissibles, afin d'en réduire le nombre. À l'aide de la figure 2.2, la méthode globale de génération des processus d'assemblage avec LEGA est illustrée. Cette méthode suit les étapes ci-dessous (voir J.M. HENRIOUD [32]) :

- introduction des contraintes stratégiques,
- détermination des opérations réalisables, et des processus admissibles,
- choix du composant de base,
- évaluation des processus d'assemblage.

Par cohérence avec les graphes de précédence qui font l'objet de la section suivante, la représentation des processus d'assemblage qui sera utilisée est la représentation par graphes d'assemblage (voir section 1.4.2.1).

2.2.3 Conclusion

Après cette rapide présentation de la méthode de génération des processus d'assemblage, nous estimons que les travaux de J.M. Henrioud [32] sont suffisants et nous ne les remettrons pas en cause, malgré une limite vite atteinte de LEGA. Nous allons étudier plus précisément trois des modes de représentation des processus d'assemblage présentés au chapitre 1.

2.3 Graphes d'assemblage

Comme nous l'avons vu dans le chapitre précédent, le graphe d'assemblage est une méthode de représentation des processus d'assemblage.

2.3.1 Représentation

Selon la définition 1.15, les nœuds représentent des opérations, et les arcs entre ces nœuds représentent eux l'enchaînement de ces opérations. À l'aide de la figure 1.7, deux grands types de graphes d'assemblage ressortent, c'est à dire :

- graphes d'assemblage linéaires ;
- graphes d'assemblage arborescents.

Le graphe d'assemblage est essentiellement un graphe de précédence qui présente une structure toute particulière. Cette structure lui permet de conserver tous les avantages de la représentation des processus d'assemblage par arbres. Comme objet mathématique, le graphe d'assemblage est une anti-arborescence.

Avant de passer à la présentation des graphes de précédence et des ASTD, nous allons poser quelques hypothèses.

2.3.2 Hypothèses de travail et notations

Nous utilisons comme données d'entrée de notre analyse, les processus d'assemblage établis par LEGA (J.M. Henrioud [31]) représentés par des graphes d'assemblage.

Afin de simplifier le problème, nous ne traiterons que les graphes d'assemblage linéaires, car un graphe d'assemblage non linéaire peut être linéarisé, en considérant le sous-assemblage obtenu par la ligne d'assemblage parallèle, comme un constituant élémentaire et donc en ne tenant pas compte de son assemblage. Par exemple avec le stylobille de la figure 1.2, si nous considérons les trois constituants (le composant *« capuchon »*, le sous-assemblage *« tête, cartouche, encre »* et le sous-assemblage *« corps, bouchon »*) alors nous réduisons la complexité du problème à un produit de trois constituants.

Pour éviter toute confusion, nous appellerons ces graphes d'assemblage linéaires, préalablement établis à l'aide de LEGA, des *séquences d'enchaînement*. L'enchaînement d'une opération α_j après une opération α_i avec i différent de j est notée : α_i — α_j.

De plus nous considérons que toutes les séquences conduisant à un même sous-assemblage ont le même composant de base. Ceci est cohérent avec le fonctionnement du logiciel LEGA qui regroupe les gammes en sous-ensembles cohérents, c'est à dire susceptibles d'être réalisées sur un même système physique, donc faisant intervenir les mêmes sous-assemblages, chacun d'entre eux ayant un composant de base donné (correspondant à un posage unique).

Dans ces conditions, chaque sous-problème considéré se ramène à considérer :
- un ensemble E de m tâches. $E = \{\alpha, \beta, \ldots, u\}$ où α est le chargement du composant de base et u le déchargement du produit fini ;
- un ensemble Υ de n séquences.

Chaque séquence comporte m tâches, chacune apparaissant évidemment une et une seule fois.

Remarque 2.1
Toutes les séquences d'enchaînement commencent par α et finissent par u.

2.3.3 Conclusion

Cette section nous a présenté les graphes d'assemblage, qui sont à l'origine de nos travaux. L'ensemble des graphes d'assemblage linéaire définis par LEGA, respectant les hypothèses ci-avant sont appelés *séquences d'enchaînement*. Nous proposons quelques rappels sur les graphes de précédence, sur les ASTD et sur les hypergraphes de précédence avant de présenter des méthodes de génération de graphes de précédence.

2.4 Les graphes de précédence

Comme il a été précisé dans le précédent chapitre, les graphes de précédence sont utilisés, la plupart du temps, en conception de ligne de production. Pour aborder la notion de graphe de précédence, nous commenterons quelques travaux sur l'élaboration des graphes de précédence, puis nous rappellerons leur définition formelle, et quelques outils de comparaison de deux graphes de précédence.

2.4.1 Représentation formelle des graphes de précédence

DÉFINITION 2.1 (GRAPHE DE PRÉCÉDENCE)
Un graphe de précédence gp est défini par un couple $< E, U >$ où :

- *E est un ensemble de* tâches *d'assemblage ;*
- *U est un ensemble de contraintes de précédence définies sur $E \times E$.*

Si nous reprenons l'exemple de la figure 1.2, les 3 graphes de précédence s'écrivent : $gp_a =< E_a, U_a >$, $gp_b =< E_b, U_b >$, $gp_c =< E_c, U_c >$ où par exemple
$E_a = \{B, H, Cr, CP, I, T, u\}$,
$U_a = \{(B, H), (H, Cp), (Cp, u), (H, Cr), (Cr, I), (I, T), (T, u)\}$

Pour une question de lisibilité, les graphes de précédence $gp =< E, U >$ sont représentés graphiquement par leur graphe partiel $gp_i =< E_i, U_i >$ où :
$\forall \alpha_1, \alpha_2 \in E, (\alpha_1, \alpha_2) \in U \Leftrightarrow (\alpha_1, \alpha_2) \in U_i \, \forall (\alpha_1, \alpha_3) \in U, (\alpha_3, \alpha_2) \notin U_i$

Avec l'exemple de la figure 1.2, nous obtenons la représentation graphique du graphe de précédence $gp_a =< E_a, U_a >$ présentée dans la figure 2.3.

REMARQUE 2.2
Puisqu'un graphe de précédence définit un ordre partiel, certaines opérations sont partiellement spécifiées : seul le composant secondaire (ou le caractère non géométrique pour une opération non géométrique) est précisé, le constituant primaire dépendant du processus (ordre total) qui sera effectivement suivi.
Dans ces conditions, nous noterons x l'ensemble ayant x comme constituant secondaire (ou comme caractère non géométrique). Et nous parlerons, par commodité et conformément à l'habitude de tâche.

Une opération d'assemblage géométrique est l'assemblage d'un constituant secondaire sur un constituant primaire immobilisé. La réalisation de cette opération est principa-

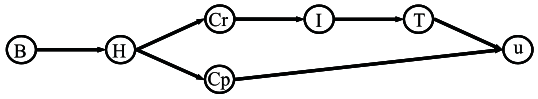


FIG. 2.3 – Graphes de précédence du stylo bille

lement conditionnée par le constituant secondaire, par ses manipulations. C'est pour cela que nous représentons une opération par son caractère ou son constituant dans les graphes d'assemblage, l'opération de chargement par le composant de base manipulé, et l'opération de déchargement par le symbole u.

DÉFINITION 2.2 (CONTRAINTE DE PRÉCÉDENCE)
Une contrainte de précédence entre deux opérations est une relation binaire décrivant le fait que l'une doit être réalisée avant l'autre. La contrainte (e_1, e_2) *signifie que la réalisation de l'opération* e_1 *doit toujours précéder celle de l'opération* e_2*, elle est aussi notée* $e_1 \rightarrow e_2$.

Selon A. DELCHAMBRE [18], une contrainte de précédence impose la réalisation d'une opération avant une autre, ceci pour des raisons de stabilité, géométrique ou technologique.

De par la définition 1.15 et les définitions 2.1, 2.2, il est possible de dire que les graphes d'assemblage sont aussi des graphes de précédence et que le graphe de précédence est une définition collective des graphes d'assemblage, qui sont eux-mêmes une traduction des arbres d'assemblage. Il existe donc un lien étroit entre les graphes d'assemblage et les graphes de précédence.

DÉFINITION 2.3 (SÉQUENCES D'ENCHAÎNEMENT ASSOCIÉES)
On appelle séquences d'enchaînement associées *à un graphe* G *représentatif des processus d'assemblage, l'ensemble des séquences d'enchaînement respectant ce graphe, noté* $SEA(G)$.

Par exemple, avec le graphe de précédence de la figure 2.3, nous avons :
$SEA(G_e) = \{B — H — Cr — I — T — Cp — u, B — H — Cr — Cp — I — T — u, B — H — Cr — I — Cp — T — u\}$

2.4.2 Comparaison de graphes de précédence

Dans la suite de ce travail, nous aurons besoin des deux relations d'ordre sur l'ensemble des graphes de précédence associés à un ensemble d'opérations, définies ci-après.

DÉFINITION 2.4 (COMPARAISON DE DEUX GRAPHES DE PRÉCÉDENCE)
Soient $gp_1 =< E_1, U_1 >$ *et* $gp_2 =< E_2, U_2 >$ *deux graphes de précédence,*

- gp_1 *est dit* égal *à* gp_2 *(*$gp_1 = gp_2$*) si* $(E_1 = E_2) \wedge (U_1 = U_2)$;
- gp_1 *est dit* inférieur[4] *à* gp_2 *(*$gp_1 < gp_2$*) si* $(E_1 = E_2) \wedge (U_1 \supset U_2)$ $/\forall (e_1, e_2) \in U_2 \Rightarrow (e_1, e_2) \in U_1$
  Cette infériorité se traduit par un nombre de gammes générées par gp_1 *inférieur au nombre de gammes générées par* gp_2.

Un graphe de précédence gp est dit *minimal* s'il est inférieur à tout autre graphe de précédence gp_i du même assemblage : $\forall gp_i, \neg(gp > gp_i)$.

Soient GP un ensemble de graphes de précédence, un graphe de précédence gp est dit *maximal* au sein de GP si pour tout gp_i de GP, nous avons gp_i inférieur à gp : $gp \in GP$ et $\forall gp_i \in GP, \neg(gp < gp_i)$.

Soient gp un graphe de précédence, une séquence d'enchaînement SE est dite représentée par un graphe de précédence gp si $SE \leq gp$.
$SEA(gp)$ étant l'ensemble des séquences d'enchaînement représentées par le graphe de précédence gp, nous avons la propriété suivante :

PROPRIÉTÉ 2.1 (INCLUSION DES SEA)
Soit $gp_1 =< E_1, U_1 >$ *et* $gp_2 =< E_2, U_2 >$ *deux graphes de précédence.*

$$gp_1 \geq gp_2 \quad \Rightarrow \quad SEA(gp_1) \supseteq SEA(gp_2)$$

DÉMONSTRATION 2.1
Soit $gp_1 =< E_1, U_1 >$ *et* $gp_2 =< E_2, U_2 >$ *deux graphes de précédence, supposons que* s *soit une séquence d'enchaînement quelconque au sein de* $SEA(gp_2)$*. Nous avons* U *telle que* $U \supseteq U_2$*. Si* $gp_1 > gp_2$*, c'est-à-dire* $U_1 \subset U_2$*, nous avons aussi* $U \supseteq U_1$*. En conséquence,* s *fait partie de* $SEA(gp_1)$

Sachant qu'un graphe de précédence décrit un ordre partiel entre des opérations, nous pouvons traduire l'absence de relation d'ordre entre deux opérations par le parallélisme ou la permutation au sein des graphes de précédence.

2.4.3 Parallélisme au sein d'un graphe de précédence

Deux constituants sont dits *indépendants* s'ils n'ont ni de constituants en commun, ni de caractères non géométriques en commun. Nous parlons de parallélisme dans le cas où

[4] gp_2 est dit aussi un graphe partiel de gp_1

nous avons deux opérations constitutives réalisées à partir de constituants indépendants construits à partir de deux composants de base différents.

Nous pouvons dire que deux opérations sont parallèles si elles n'ont aucun prédécesseur commun. Par exemple, avec le stylo-bille de la figure 1.2, si nous prenons le sous-assemblage « tête, encre, cartouche » et le sous-assemblage « bouchon, corps », ils n'ont aucun constituant en commun, alors ces deux sous-assemblages peuvent être établis en parallèle.

2.4.4 Élaboration des graphes de précédence

Un grand nombre de travaux[5] traite de la génération des graphes de précédence. Généralement, les auteurs proposent une même définition formelle des graphes de précédence, et ils ont les mêmes difficultés de formalisation et de génération de ces graphes. Ces travaux peuvent être regroupés sous trois grandes tendances : la génération des graphes de précédence à partir du *produit avec une recherche des contraintes de précédence*, ou à partir des *contraintes de précédence déjà établies* ou alors à partir des *processus d'assemblage déjà établis.*

2.4.4.1 Recherche des contraintes de précédence

Dans des travaux comme ceux de B. Frommherz et J. Hornberger [25] par exemple, les contraintes de précédence sont déduites des modèles CAD du produit à l'aide de l'approche de désassemblage. Cette méthode donne un arbre de graphes de précédence, où les branches traduisent une disjonction dans les graphes de précédence. Avec l'application de la règle d'ajout de contraintes de précédence (Stabilité, Non pénétration . . .), les branches invalides ne sont pas développées (graphes cycliques). Les contraintes introduites sont de type :

$$\alpha \rightarrow \alpha_1 \text{ et } \alpha_2 \rightarrow \alpha_3 \tag{2.1}$$

Avec l'ajout de ces quelques contraintes, il est possible de réduire le nombre de graphes de précédence, mais sans être sûr de l'exhaustivité de cette génération. De plus une deuxième règle permet de sélectionner un seul et unique graphe de précédence afin de le retenir.

[5] T.O. Prenting [46], B. Frommherz [25], A. Delchambre [18], E. Bampis [1], K. Chen [12], K.S. Naphade [42], J. Danloy [13], T.I. Liu [37], P. De Lit [15], A. Bratcu [34], [40], [11], P. Fouda [24], [22], [23] . . .

A. DELCHAMBRE [17], quant à lui, propose un *planificateur assisté par ordinateur* pour la génération de graphes de précédence. Après une séparation du modèle géométrique du produit et des données non-géométriques, les contraintes de précédence sont générées par apport des contraintes géométriques, mécaniques, technologiques et de stabilité. L'auteur introduit une notion d'*ordre de précédence*, c'est à dire qu'il décrit chaque liaison entre les composants de la manière suivante :

$$\text{ordre\_de\_précédence}([N],[L],[G]) \tag{2.2}$$

avec N le numéro de la liaison traitée, L la liste des composants appartenant à la liaison N, G la liste des composants gênant la réalisation de cette liaison.

Avec l'exemple de la figure 1.4, nous avons l'ordre de précédence suivante :

$$\text{ordre\_de\_précédence}([2],[\{H,I,Cr\},\{B,T\}],[Cp]) \tag{2.3}$$

L'auteur a introduit ultérieurement dans [18], la notion de contrainte disjonctive de précédence, de type :

$$\text{doit\_précéder}(a \text{ ou } c, b) \tag{2.4}$$

où a, b et c sont trois tâches d'assemblage. Nous avons donc la tâche a ou la tâche c doit précéder la tâche b.

Avec ces déterminations des contraintes de précédence, les graphes de précédence générés sont corrects, mais ce planificateur ne permet pas de savoir si toutes les solutions sont présentes dans le résultat. L'ensemble des graphes de précédence n'est pas généré avec cette méthode.

Dernièrement, une majeure partie des travaux développés, comme ceux de J. DANLOY [13], P. DE LIT [15] ou P. FOUDA [21], [22], [23], [24] par exemple, traitent de la génération de graphes de précédence pour les *familles de produits*.

P. DE LIT [15] a utilisé une méthode proche de celle de B. FROMMHERZ et J. HORNBERGER [25], qui consiste à générer un arbre des solutions possibles à l'aide des contraintes de précédence, et ensuite à supprimer l'ensemble des branches générant des graphes non valides. Cette méthode est appliquée sur la génération de gammes d'assemblage pour les familles de produits, elle permet de générer un ensemble de graphes de précédence valides mais il n'est pas possible de savoir si tous les graphes de précédence sont générés.

Nous voyons dans le travail de J. DANLOY [13], que l'auteur a été inspiré des travaux de G. DINI [19] en ce qui concerne la notion de *produit* et de *formalisme*, tout comme P. FOUDA [21]. Ce dernier introduit la notion de *matrice d'interférence généralisée* afin

de mieux concrétiser les interactions entre les composants en assemblage. Il appelle cette matrice : $I_{(dt)}$, dt étant la direction d'assemblage. Après un ensemble d'actualisation des matrices, chaque fois qu'un composant peut être désassemblé, un graphe de précédence est obtenu. Ce graphe de précédence est basé sur le composant de base choisi par l'expert. Nous pouvons dire que ces différentes approches sont performantes, mais pas exhaustives, car seul un nombre restreint de graphes de précédence est généré.

2.4.4.2 Contraintes de précédence déjà établies

K.S. Naphade [42], dans ses travaux, a développé une méthode de génération systématique de l'ensemble des graphes de précédence correspondant à un ensemble donné de contraintes de précédence. K.S. Naphade a introduit la notion de problème deux-satisfait (2-STA)[6] pour obtenir un ensemble de partitions. Ce qui revient à découper le problème en un ensemble de sous-problèmes simples à résoudre. Puis après cette décomposition, la définition suivante est apportée pour les contraintes disjonctives :

$$(\neg p + q) \Leftrightarrow (p \Rightarrow q) \tag{2.5}$$

À l'aide de cette équivalence, les disjonctions sont transformées en équivalences, et la négation d'une précédence apparaît. L'auteur, de cet ensemble d'équivalences, va générer trois partitions appelées respectivement I, I_0, I_c, avec la propriété de partitionnement B.1 développée dans l'annexe B, à partir desquelles un ensemble de contraintes de précédence est généré. Cet ensemble permet la génération des graphes de précédence dans les cas favorables.

L'auteur, par une définition « *incorrecte* » du complémentaire de *a précède b*, peut générer des *graphes cycliques de précédence*, qui en fait ne sont pas des graphes de précédence, car d'après la définition 2.1 p. 32, un graphe de précédence est non cyclique.

Suite aux travaux de K.S. Naphade, P. De Lit [15] a tenté, sans succès, d'améliorer la méthode de génération des graphes de précédence (voir annexe B).

2.4.4.3 À partir des processus d'assemblage

K. Chen [12] montre les insuffisances des travaux qu'il a analysés, comme par exemple l'imprécision de la définition de l'*opération d'assemblage*, ou la relative maladresse du

[6] Two-satisfability : Une expression P est dite 2-SAT si et seulement si elle est constituée d'un ensemble de conjonctions de clauses de précédences où ces dernières sont soit des contraintes de précédence, soit des disjonctions de deux contraintes de précédence.

passage direct des contraintes d'assemblage aux contraintes de précédence, qui pose un problème lorsque les contraintes d'assemblage ne sont pas des relations binaires. K. CHEN génère l'ensemble des graphes de précédence maximaux par une méthode indirecte à partir d'un ensemble des graphes d'assemblage préétablis. Cette méthode part d'une base de deux hypothèses : les contraintes matérielles ne sont pas introduites dans l'élaboration des arbres d'assemblage, et les arbres d'assemblage élaborés n'ont pas subit de sélection à posteriori, et comme le dit l'auteur lui-même, la résolution du problème posé est complexe dans son principe. Sa complexité n'étant pas calculée et en l'absence de maquette logicielle, il est difficile d'évaluer son efficacité. Nous pouvons quand même conclure que c'est une méthode très difficile à mettre en place.

A. BRATCU [11], quant à elle, a une approche différente, comme K. CHEN, sa méthode débute avec l'ensemble des gammes d'assemblage établies à l'aide du logiciel LEGA. L'auteur établit une propriété (propriété Π) que doit posséder un ensemble de gammes pour qu'il lui corresponde biunivoquement un graphe de précédence. D'où la méthode utilisée qui consiste à découper l'ensemble des gammes initial en sous-ensembles possédant la propriété Π puis en déterminant pour chacun d'eux le graphe de précédence correspondant. Cette méthode est efficace, cependant la complexité de l'algorithme de découpage de l'ensemble en sous-ensembles représentables par un graphe de précédence unique n'est pas estimée, mais nous pouvons penser qu'elle n'est pas polynômiale.

2.4.4.4 Synthèse

Grâce à ce tour d'horizon, nous voyons qu'il n'existe pas de méthode simple et efficace de génération des graphes de précédence. La plupart des auteurs propose des méthodes de génération d'un seul graphe ou d'une partie seulement des graphes de précédence possibles. À l'heure actuelle, il n'existe pas encore de procédé permettant de dire si la *séquence d'enchaînement optimale*[7] est ou n'est pas dans un graphe donné, et donc nous ne savons pas si la génération des autres graphes est nécessaire. Nous ne savons pas non plus si cette non génération des graphes de précédence a des conséquences sur les méthodes de conception de système. Alors après avoir présenté un ensemble de définitions et de rappels, nous proposerons une méthode de génération exhaustive des graphes de précédence dans les prochains chapitres, mais avant cela, nous devons mieux connaître cette représentation des processus d'assemblage.

[7] Séquence d'enchaînement dont le coût d'investissement, ou le coût de fonctionnement, ou le temps de production est moindre

2.5 Passage d'un modèle d'un produit aux contraintes de précédence

Sachant que le modèle du produit établit des relations entre les composants, et que les contraintes de précédence décrivent des relations entre les opérations de production, nous pensons que la génération directe peut être mise en doute. Selon J.M. HENRIOUD [30], l'établissement des contraintes d'assemblage est très complexe. De plus, il n'y a pas de relation directe entre les contraintes d'assemblage et les contraintes de précédence. Ces contraintes d'assemblage sont principalement de deux types :

- les *contraintes opératoires* imposées par le produit lui-même, donc indépendantes du système de production, elles se classent en trois groupes :
  1. s'il est impossible de réaliser l'opération d'assemblage (S, α_i) sur le sous-assemblage S. Cette impossibilité peut-être de plusieurs types, comme par exemple, un constituant sur la trajectoire de l'opération à réaliser, inaccessibilités diverses. Cette contrainte est dite *contrainte géométrique* ;
  2. s'il est impossible de maintenir le constituant C suivant au moins une orientation, de telle façon, qu'en l'absence de mouvement, toutes les liaisons géométriques restent établies sous l'action des forces potentielles agissant sur lui. Cette contrainte est dite *contrainte de stabilité* ;
  3. s'il n'y a ni contrainte géométrique, ni contrainte de stabilité, si l'opération d'assemblage(S, α_i) est impossible ou trop difficile à réaliser, nous dirons qu'il y a une *contrainte matérielle*. Elle dépend de l'évaluation de l'expert.
- les *contraintes stratégiques* émises par un expert humain, dues aux particularités morphologiques du produit. Elles imposent certains sous-assemblages, ou elles imposent de regrouper certaines opérations devant être effectuées dans une même direction. Pour les familles de produits, nous cherchons à obtenir des gammes associées aux différents objets de la famille aussi voisines que possible.

Les contraintes de précédence sont principalement de deux types :

- stabilité,
- pénétration.

Nous pouvons craindre que l'absence de relation entre les contraintes d'assemblage et les contraintes de précédence, nous pousse à écarter la possibilité de génération directe des graphes de précédence.

2.5.1 Conclusion

Nous estimons que ces travaux menés sur la génération des graphes de précédence sont insuffisants. Aujourd'hui encore, le moyen le plus utilisé pour la génération des graphes de précédence d'un produit, est la génération par un expert humain. Ce qui est encore acceptable pour un produit *simple* ou sur une ligne de production existante, mais dans le cas d'un produit *complexe*[8] ou de la création d'une nouvelle ligne de production, même un expert habile ne peut être sûr que le graphe de précédence difficilement trouvé est le plus pertinent. Rien ne permet de dire, aujourd'hui, si un graphe de précédence est plus efficient qu'un autre. Nous devons donc générer tous les graphes de précédence puis en faire une sélection, en évaluant les résultats obtenus pour chacun d'eux.

Notre objectif est d'obtenir un système optimal et d'offrir la plus grande flexibilité possible. Pour cela, nous choisissons de travailler sur la génération des graphes de précédence d'assemblage. Afin de garantir l'obtention d'un système optimal, il est nécessaire que la séquence d'enchaînement optimale appartienne aux graphes de précédence générés. La seule manière actuelle d'être certain de l'appartenance de cette solution à un graphe de précédence ou ensemble de graphes de précédence d'un produit, est la génération de *tous* les graphes de précédence d'assemblage maximaux admissibles du produit en question. Nous consacrerons les prochains chapitres à chercher des solutions de génération exhaustives des graphes de précédence, à partir d'un ensemble de séquences d'enchaînement prédéfinies. Dans nos travaux [48], nous avons établi une possibilité de transformation des séquences d'enchaînement en graphes de précédence (voir chapitre 3) en passant par les ASTD. Cette idée est née, de l'observation des similitudes entre les graphes de précédence et les graphes d'état (ASTD). La représentation des processus par les ASTD est exposée ci-après, afin d'apporter les connaissances suffisantes à la compréhension de notre méthode, exposée au chapitre 3.

2.6 ASTD

Les *ASTD*[9] ont été présentés brièvement dans la section 1.4.2.5. Compte-tenu de leur intérêt pour la suite, nous en faisons ci-après une analyse plus détaillée. La notion d'ASTD a été introduite à la fin des années 1980 par C.J.M. Heemskerk [27]. Cette modélisation est basée sur la représentation des états du produit intermédiaire. D'après

[8] Produit composé de plus d'une douzaine de constituants

[9] Assembly State Transition Diagram *ou* Graphes d'état des transitions d'assemblage de simple niveau, *appelé aussi* Graphes d'état

A. Bourjault [10], nous appelons *état du produit intermédiaire* l'ensemble des liaisons déjà établies à une étape quelconque du processus d'assemblage.

2.6.1 Introduction

Les ASTD ont été introduits comme une représentation adéquate pour les gammes d'assemblage. Cette représentation est basée sur l'assemblage linéaire des produits. Dans sa forme complète, l'*ASTD* représente toutes les séquences possibles d'assemblage d'un groupe de composants. Un ASTD est un graphe direct composé de nœuds et d'arcs et ne possédant pas de cycle.

Chaque chemin allant du nœud origine au nœud final est une séquence opératoire complète. De plus, chaque état comporte autant d'éléments que le produit comporte de constituants.

2.6.2 Quelques travaux

La littérature portant sur les ASTD et leurs extensions est relativement réduite, un nombre restreint d'auteurs a travaillé sur cette méthode de représentation. A notre connaissance, seuls C.J.M. Heemskerk, N. Boneschancher et leurs collègues, [5], [6], [7], [27], [28], [29] et L. Relange et J.M. Henrioud [48], [49] ont travaillé sur cette représentation. Dans ces approches, trois méthodes de génération ont été introduites : une génération directe par C.J.M. Heemskerk et N. Boneschancher, et deux générations indirectes par L. Relange et J.M. Henrioud.

2.6.3 Génération

Très peu d'auteurs se sont attachés à la génération des ASTD et à leur amélioration. C.J.M. Heemskerk au début des années 90 avec ses collègues et principalement N. Boneschanscher dans les années 89-93 ont travaillé sur ce sujet, dans leurs travaux [5], [7], [28] et [29] ils ont établi une méthode de génération des LASTD. Cette méthode est d'un principe simple, mais relativement lourde à exécuter. Elle consiste à générer tous les états intermédiaires théoriques possibles du produit (valides et non valides). La seconde phase de cette méthode est la suppression de tous les états non stables, ou inaccessibles, ainsi que toutes les transitions non valides. Cette phase est appelée l'*élagage des LASTD*.

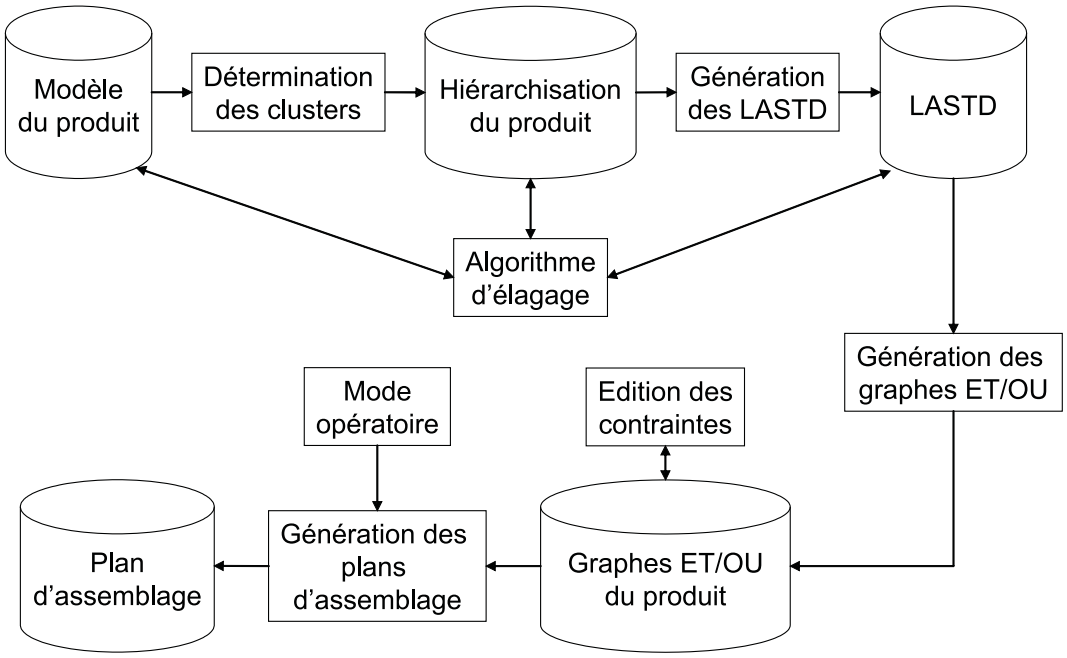
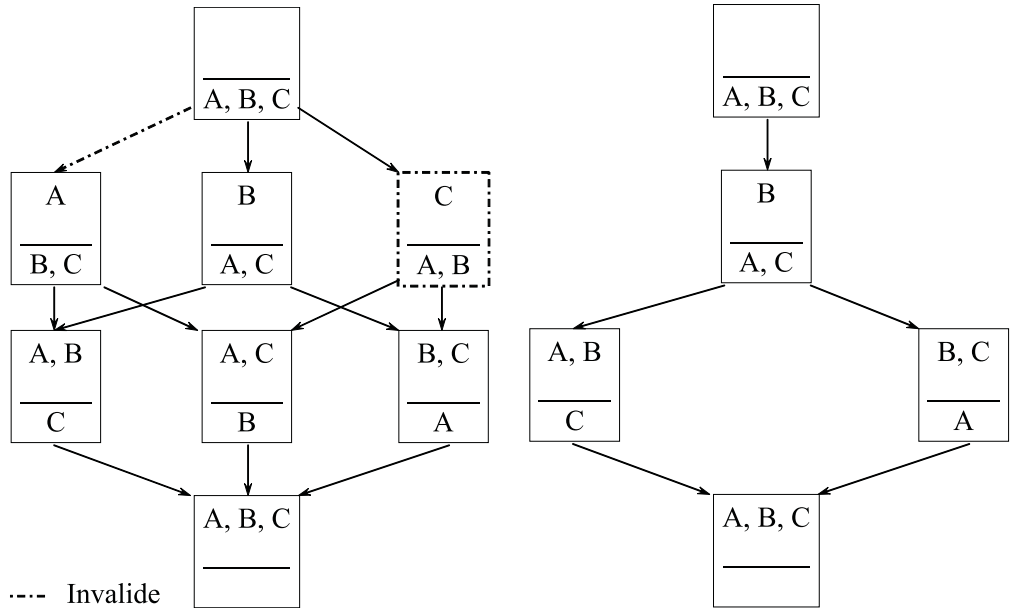


FIG. 2.4 – Les différentes étapes de la génération des plans d'assemblage

Cette phase d'élagage est basée sur le modèle du produit, c'est-à-dire en fonction des contraintes opératoires et stratégiques du produit. En *hiérarchisant* ces propriétés, il arrive de manière adroite à générer les LASTD correspondant au produit après un certain nombre d'élagages au niveau du modèle du produit, de la hiérarchisation, et pour finir des LASTD. Ces méthodes d'élagage sur la stabilité et les transitions invalides sont trop complexes pour être développées ici, elles peuvent être trouvées dans la thèse de C.J.M. HEEMSKERK [29]. Cette méthode d'élagage est implantée dans la méthode de génération de N. BONESCHANSCHER, de plus elle peut être appliquée de deux manières : automatique ou manuelle, afin de mieux gérer les séquences générées. L'auteur, à l'aide d'outils et d'un ensemble de données, transforme les LASTD en graphes ET/OU pour déterminer les plans d'assemblage, afin de faire intervenir des contraintes ou de les modifier, puis il détermine les plans d'assemblage depuis ces graphes ET/OU. Nous illustrons cette démarche à l'aide de la figure 2.4.

Comme nous l'avons vu, cette génération directe est composée de deux grandes parties, une génération de tous les états possibles, puis un élagage de tout état non stable, non admissible, et de toute transition non valide. Le processus d'*élagage*, introduit par C.J.M. HEENSKERK, est un moyen de réduire le nombre de séquences en supprimant les transitions invalides et les états instables ou inaccessibles. Cet élagage est exposé à



(a) ASTD complet (b) ASTD élagué

FIG. 2.5 – Élagage d'un ASTD

la figure 2.5. La figure 2.5(a) montre tous les états possibles d'un assemblage de trois constituants, mais la transition de l'état $\bar{A}, \bar{B}, \bar{C}$ à l'état $A, \bar{B}, \bar{C}$ est invalide, ainsi que l'état $\bar{A}, \bar{B}, C$. Comme la transition de l'état $\bar{A}, \bar{B}, \bar{C}$ à l'état $A, \bar{B}, \bar{C}$ est invalide, l'état $A, \bar{B}, \bar{C}$ est inaccessible alors toute transition partant de cet état est invalide, les transitions de l'état $A, \bar{B}, \bar{C}$ à l'état $A, B, \bar{C}$ et de l'état $A, \bar{B}, \bar{C}$ à l'état $A, \bar{B}, C$ sont invalides. De plus comme l'état $\bar{A}, \bar{B}, C$ est invalide, toute transition lui étant incidente est invalide, et toute transition partant de cet état est invalide. Les transitions de l'état $\bar{A}, \bar{B}, \bar{C}$ à l'état $\bar{A}, \bar{B}, C$, de l'état $\bar{A}, \bar{B}, C$ à l'état $\bar{A}, B, C$ et de l'état $\bar{A}, \bar{B}, C$ à l'état $A, \bar{B}, C$ sont alors supprimées. La suppression de ces transitions implique la non accessibilité de l'état $A, \bar{B}, C$, ce qui implique sa suppression et la suppression de la transition de l'état $A, \bar{B}, C$ à l'état final A, B, C. Après cet élagage, l'ASTD de la figure 2.5(b) est alors obtenu. Cette génération a pour point de départ le produit à assembler avec un ensemble de connaissances du type : « impossibilité d'effectuer l'opération A avant l'opération B ou l'opération C » ou « instabilité de l'état $\bar{A}, \bar{B}, C$ ». Cet ensemble de connaissances sur le produit est nécessaire pour générer ces ASTD ne représentant que des processus d'assemblage admissibles.

La génération indirecte est basée sur un ensemble de séquences d'enchaînement admissibles déjà établies par le logiciel « LEGA ». Cette transformation permet de générer

directement les ASTD ne représentant que les séquences d'enchaînement admissibles. Cette génération sera développée dans la section 3.3.1 page 52.

2.6.4 Complexité

Le problème majeur d'un ASTD est sa taille. Le nombre de nœuds est directement lié au nombre N de constituants du produit ainsi qu'à sa *complexité d'assemblage*. Dans le cas le plus simple, avec seulement une séquence de production, chaque niveau n'a qu'un nœud, le graphe comporte $N + 1$ nœuds. Par contre s'il y a plusieurs séquences d'enchaînement, le nombre de nœuds peut augmenter de façon exponentielle avec le nombre de constituants.

D'après C.J.M. Heemskerk [29] et N. Boneschanscher[5], la complexité peut s'écrire ainsi, avec un produit composé de N constituants :

- si le produit considéré n'a pas de sous-assemblage, alors il y a au maximum 2^N nœuds possibles ;
- si il a un composant de base unique, le nombre d'états devient alors égal à $1+2^{N-1}$;
- si le produit considéré est composé de N constituants divisible en un sous-assemblage de m constituants et d'un sous-assemblage primaire avec p constituants, le nombre d'états de l'ASTD est : $1 + \sum_{i=0}^{p-1} C_i^{p-1} + 1 + \sum_{i=0}^{m-1} C_i^{m-1} = 2^{p-1} + 2^{m-1} + 2$ où $N = p + m - 1$.

Il est raisonnable de dire que le nombre de nœuds d'un ASTD est de l'ordre de 2^N.

2.6.5 Conclusion

Nous pouvons dire, après cette étude, que les ASTD et leurs améliorations ne sont pas très pertinents tant au point de vue de leur génération qu'au point de vue de l'élaboration des plans d'assemblage. Un point positif ressort clairement de cette représentation, c'est l'expression claire des états intermédiaires. Cette qualité, bien que peu pertinente en conception, pourrait être très utile en pilotage de système. Notre objectif étant la génération de graphes de précédence, nous ne pensons utiliser les ASTD que comme représentation intermédiaire.

2.7 Les hypergraphes de précédence

J.M. Henrioud [33] a déjà introduit la notion d'*hypergraphe de précédence* pour exprimer des contraintes disjonctives de précédence et pour réduire le nombre de graphes de précédence.

Définition 2.5 (Hypergraphe selon C. Berge [4])
Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini, et $\xi = (E_i / i \in I)$ une famille de parties de X. On dira que ξ constitue un hypergraphe *sur X si l'on a :*

$$E_i \neq \emptyset \qquad (i \in I)$$

$$\bigcup_{i \in I} E_i = X$$

Le couple $H = (X, \xi)$ s'appelle un hypergraphe, et son ordre est $|X| = n$; les éléments de X sont les sommets de l'hypergraphe, et les ensembles de ξ, que l'on note $E_1, E_2, \ldots, E_m$ sont les arêtes de l'hypergraphe.

Définition 2.6 (Hypergraphe de précédence)
Un hypergraphe de précédence *est un hypergraphe (X, ξ) où X est l'ensemble des tâches et ξ l'ensemble des hyperarcs E_k, qui ont l'une des formes suivantes :*
$E_k = (x_i, \{x_{i_1}, \ldots, x_{i_n}\})$ *si et seulement si* $x_i \rightarrow x_{i_1} + \ldots + x_{i_n}$
ou $E_k = (\{x_{i_1}, \ldots, x_{i_n}\}, x_i)$ *si et seulement si* $x_{i_1} + \ldots + x_{i_n} \rightarrow x_i$

Les hyperarcs de précédence représentent les contraintes de précédences disjonctives, par exemple, $(\alpha_1 \rightarrow \alpha_2 + \alpha_3 \rightarrow \alpha_2)$ est équivalant à $(\alpha_1 + \alpha_3) \rightarrow \alpha_2$. Cette notion est représentée dans la figure 2.6, l'arc de cercle, centré sur α_2, entre les arcs (α_1, α_2) et (α_3, α_2), définit un choix entre les deux relations de précédence. Cet hypergraphe est noté : $H = (X, \xi)$ $X = \{\alpha_1, \alpha_2, \alpha_3\}$ $\xi = \{(\{\alpha_1, \alpha_3\}, \{\alpha_2\})\}$

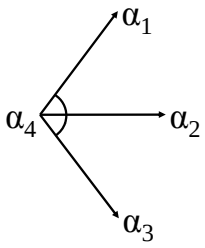
L'emploi de ces hypergraphes permet une représentation compacte alors que la présence de contraintes disjonctives nécessiterait l'emploi de plusieurs graphes de précédence.

La figure 2.7 présente quelques exemples d'hypergraphes simples. La figure 2.7(a) présente un *hyperarc* de précédence traduisant l'expression suivante :

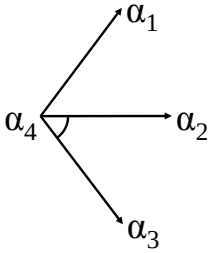
$$(\alpha_4 \rightarrow \alpha_1) + (\alpha_4 \rightarrow \alpha_2) + (\alpha_4 \rightarrow \alpha_3) \tag{2.6}$$

L'hypergraphe de précédence H_1 correspondant s'écrit :
$H_1 = (X, \xi_1)$ avec $X = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ et $\xi_1 = \{E_1\}$ $E_1 = (\{\alpha_4\}, \{\alpha_1, \alpha_2, \alpha_3\})$

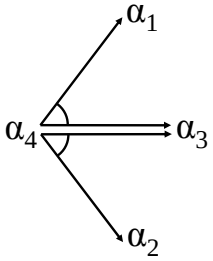
FIG. 2.6 – Exemple d'hyperarc de précédence



(a) Hyperarc



(b) Conjonction d'un hyperarc et d'un arc simple



(c) Autre conjonction de deux hyperarcs

FIG. 2.7 – Des hypergraphes de précédence

Les figures 2.7(b) et 2.7(c) présentent des *conjonctions*[10] *d'hyperarc*. La figure 2.7(b) montre un hypergraphe de précédence H_2 traduisant l'expression suivante :
$(\alpha_4 \rightarrow \alpha_1).((\alpha_4 \rightarrow \alpha_2) + (\alpha_4 \rightarrow \alpha_3))$

Cet hypergraphe de précédence s'écrit :
$H_2 = (X, \xi_2)$ avec $X = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ et $\xi_2 = \{E'_1, E'_2\}$ $E'_1 = (\{\alpha_4\}, \{\alpha_2, \alpha_3\})$
$E'_2 = (\{\alpha_4\}, \{\alpha_1\})$

La figure 2.7(c) montre un hypergraphe de précédence H_3 traduisant l'expression suivante :

$$((\alpha_4 \rightarrow \alpha_1) + (\alpha_4 \rightarrow \alpha_3)).((\alpha_4 \rightarrow \alpha_2) + (\alpha_4 \rightarrow \alpha_3)) \tag{2.7}$$

Nous écrivons, à l'aide d'un hypergraphe H_3, l'expression (2.7) de la manière suivante :
$H_3 = (X, \xi_3)$ avec $X = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ et $\xi_3 = \{E''_1, E''_2\}$ $E''_1 = (\{\alpha_4\}, \{\alpha_1, \alpha_3\})$
$E''_2 = (\{\alpha_4\}, \{\alpha_2, \alpha_3\})$, et nous le représentons comme le montre la figure 2.7(c).

Après cette extension du concept de *graphe de précédence*, nous allons proposer dans les prochains chapitres des méthodes de transformation d'un ensemble de séquences d'enchaînement en graphe de précédence ou en hypergraphes de précédence.

[10] Opérateur binaire correspondant à un «et» logique.

2.8 Conclusion du chapitre

Après avoir précisé, dans ce chapitre, l'objectif de nos travaux et rappelé un ensemble de définitions, nous avons développé des propriétés sur les représentations des processus d'assemblage. Nous avons considéré ici trois grands types de représentations des processus d'assemblage ; les *graphes d'assemblage*, les *graphes de précédence*, les *ASTD*.

Nous avons rappelé une représentation formelle du graphe de précédence. Comme nous avons pu le voir au cours de ces deux chapitres, la génération des graphes de précédence est empirique. Actuellement, il existe un certain nombre de méthodes de génération, soit délicates à mettre en place, soit non exhaustives, voire non valides pour certaines.

De là, la problématique abordée dans ce travail : établir une méthode de génération simple et efficace des graphes de précédence et de manière systématique.

La recherche d'une nouvelle méthode de génération systématique des graphes de précédence simple et efficace, nous a poussé à exposer l'ensemble des connaissances sur les graphes d'état.

Afin de permettre une représentation biunivoque des graphes d'assemblage par les graphes de précédence, nous avons introduit un ensemble d'hypothèses.

La méthode de génération systématique des graphes de précédence proposée ici est basée sur des transformations de graphes. C'est une méthode par *transformation graphique*. De plus celle-ci nous a amené à introduire les hypergraphes de précédence, qui sont une évolution des graphes de précédence, car elle permet de les engendrer. Ces hypergraphes de précédence permettent de représenter un ensemble de graphes de précédence en un seul graphe, sans toutefois remettre en cause la lisibilité des processus d'assemblage.

Pour l'ensemble de ces raisons, nous utiliserons le chapitre suivant à étudier cette nouvelle méthode de génération des graphes de précédence.

Chapitre 3

Détermination des graphes de précédence par transformation de graphes

Dans le chapitre précédent, nous avons avec l'état de l'art présenté que les quelques méthodes existantes pour la détermination des graphes de précédence présentaient toutes, à des degrés divers, un certain nombre d'insuffisances ou de difficultés. C'est pourquoi une nouvelle approche sera proposée dans ce chapitre. Cette méthode de génération est une méthode permettant de générer les graphes de précédence, à partir des séquences d'enchaînement par observations et transformations de graphes. Pour ce faire nous passons par les étapes suivantes :

- transformation de l'ensemble des graphes d'assemblage en un ASTD ;
- transformation de l'ASTD obtenu en son graphe dual, le graphe d'enchaînement ;
- transformation du graphe d'enchaînement en un graphe ou hypergraphe de précédence.

Avant de présenter cette méthode, quelques points sur les méthodes de représentation des processus d'assemblage utilisés doivent être précisés.

Les graphes d'assemblage, vérifiant un certain nombre d'hypothèses, comme nous l'avons vu à la section 2.3.2, sont appelés des *séquences d'enchaînement*. Après un apport sur les ASTD et l'introduction des graphes d'enchaînement, un développement de la méthode de génération des graphes de précédence sera proposé.

3.1 ASTD

Suite à la présentation des ASTD faite dans le précédent chapitre, il ressort qu'une représentation formelle manque à cette méthode de représentation des processus d'assemblage. Afin de faciliter la manipulation des ASTD, une représentation formelle, qui est inspirée de celle des graphes de précédence, est introduite :

Définition 3.1 (ASTD)
Un ASTD *est défini par un couple* $< \xi, \tau >$ *où :*

- ξ *est l'ensemble des états admissibles ;*
- τ *est l'ensemble des opérations, permettant de passer d'un état* i *à un état* j*, définies sur* $\xi \times \xi$*.*

Une opération entre deux états est une relation binaire décrivant l'action d'assemblage d'un constituant ou d'un caractère non géométrique avec un sous-assemblage précis déjà existant.

Par exemple, l'ASTD élagué $\mathcal{A}$ de la figure 2.5(b) page 43 se note : $\mathcal{A} =< \xi, \tau >$ où
$\xi = \{< \bar{A}, \bar{B}, \bar{C} >, < \bar{A}, B, \bar{C} >, < A, B, \bar{C} >, < \bar{A}, B, C >, < A, B, C >\}$ et
$\tau = \{(< \bar{A}, \bar{B}, \bar{C} >, B), (< \bar{A}, B, \bar{C} >, A), (< \bar{A}, B, \bar{C} >, C), (< A, B, \bar{C} >, C),$
$(< \bar{A}, B, C >, A)\}$

3.2 Graphes d'enchaînement

Un graphe d'enchaînement est le *graphe dual* d'un ASTD.

Définition 3.2 (Graphe dual)
Un graphe G' *est dit dual d'un graphe* G *si à chaque arc de* G *correspond un sommet dans* G' *et si à chaque sommet de* G *correspond un arc dans* G'*.*

Dans un même souci de performance et de simplicité que pour les ASTD, une représentation formelle des graphes d'enchaînement est introduite :

Définition 3.3 (Graphe d'enchaînement)
Un graphe d'enchaînement G_D *d'un ASTD* $\mathcal{A} =< \xi, \tau >$ *est défini par le 4-uplet* $< E, \sigma, \xi, L >$ *où :*

- E *est l'ensemble des tâches appartenant à* τ *;*
- σ *est l'ensemble des successions directes des tâches réalisables de l'ensemble* E*, définies sur* $E \times E$ *;*

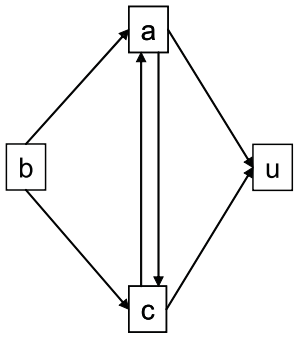


FIG. 3.1 – Graphe d'enchaînement de l'ASTD de la figure 2.5(b)

- *ξ est l'ensemble des états admissibles de l'ASTD ;*
- *L est une application de σ dans ξ.*

Par exemple, le graphe d'enchaînement de l'ASTD élagué $\mathcal{A}$ de la figure 2.5(b) page 43, donne le graphe d'enchaînement G_D suivant :
$G_D =< E, \sigma, \xi, L >$ où
$E = \{a, b, c, d, u\}$,
$\sigma = \{(b, a), (b, c), (a, c), (c, a), (c, u), (a, u)\}$,
$\xi = \{< \bar{A}, \bar{B}, \bar{C} >, < \bar{A}, B, \bar{C} >, < A, B, \bar{C} >, < \bar{A}, B, C >\}$
et $L = \{((b, a), < \bar{A}, \bar{B}, \bar{C} >), ((b, c), < \bar{A}, \bar{B}, \bar{C} >), ((a, c), < \bar{A}, B, \bar{C} >),$
$((c, a), < \bar{A}, B, \bar{C} >), ((c, u), < A, B, \bar{C} >), ((a, u), < \bar{A}, B, C >)\}$,
qui est représenté à l'aide la figure 3.1.

Après ces définitions, notre méthode de génération des graphes de précédence à partir de l'ensemble des séquences d'enchaînement pré-établies avec LEGA sera présentée.

3.3 Méthode proposée

N'ayant toujours pas de méthode de génération systématique des graphes de précédence simple et efficace, notre intérêt s'est porté plus spécialement sur les graphes d'état des transitions d'assemblage (ASTD), qui pour nous, ont un intérêt, dans cette génération, de par leur proximité avec les graphes de précédence. Voyant que l'ASTD représentant un processus d'assemblage d'un produit, dans certain cas simple, est le graphe dual du graphe de précédence de ce même processus, notre méthode de génération des graphes de précédence est développée à partir de cette idée.

Cette méthode par transformation de graphes[1], est composée de trois phases distinctes. Ces phases sont *la génération d'un ASTD* à partir d'un ensemble Υ de séquences d'enchaînement (notée $\mathcal{M}_{\mathcal{A}}(\Upsilon)$), *le passage au graphe d'enchaînement* de l'ASTD (noté $\mathcal{M}_D(\Upsilon)$) et *la génération du graphe de précédence* à partir du graphe d'enchaînement (noté $\mathcal{M}_G(\Upsilon)$). Ces trois phases seront développées successivement.

3.3.1 Génération d'un ASTD

Un ASTD est la représentation collective des séquences d'enchaînement de l'ensemble Υ. La génération de cet ASTD est constituée de deux phases récurrentes ; la première est la détermination de l'état accessible suivant, la deuxième est la détermination de l'arc, qui représente la tâche, liant l'état courant à l'état accessible suivant. Ces deux phases sont répétées pour chaque tâche de chaque séquence d'enchaînement de l'ensemble Υ. De plus nous appelons sous-séquence une suite de tâches, d'une séquence x_i, partant soit de la tâche de chargement jusqu'à une tâche α_i quelconque antérieure à la tâche u de déchargement, si cette sous-séquence comprend la tâche α_i, elle est notée ${x_i}_{\alpha]}$[2], sinon elle est notée ${x_i}_{\alpha[}$[3], soit de la tâche α_i postérieure à la tâche de chargement jusqu'à la tâche u de déchargement, si cette sous-séquence comprend la tâche α_i, elle est notée ${x_i}_{[\alpha}$[4], sinon elle est notée ${x_i}_{]\alpha}$[5]. Il est nécessaire d'introduire la définition 3.4 de l'*état engendré* pour écrire l'algorithme 3.1.

Définition 3.4 (État engendré)
On appelle état engendré, *l'état obtenu par une sous-séquence d'enchaînement donnée* x.
Il est noté :

$$\forall x, \mathcal{E}(x)$$

L'algorithme 3.1 de génération d'un ASTD à partir d'un ensemble de séquences d'enchaînement, transcrit ce procédé de transformation. Pour une tâche donnée d'une séquence d'enchaînement donnée, à l'aide de l'état courant, l'état accessible est généré. Si l'état accessible n'est pas encore dans l'ASTD, il est ajouté dans l'ensemble ξ des états admissibles. Puis la tâche de l'état courant est ajoutée à l'état accessible dans l'ensemble τ, si celle-ci n'existe pas encore. L'état accessible ainsi généré devient l'état courant, et

[1] Méthode basée sur l'évolution de différents types de graphes
[2] Notion d'ensemble : $[\alpha, \alpha_i]$
[3] Notion d'ensemble : $[\alpha, \alpha_i[$
[4] Notion d'ensemble : $[\alpha_i, u]$
[5] Notion d'ensemble : $]\alpha_i, u]$

Entrée : Υ Ensemble des séquences d'enchaînement
Variable : état courant, état obtenu, état initial
Sortie : ASTD
  Pour chaque séquence d'enchaînement de Υ **faire**
    Pour chaque tâche **faire**
      // Traiter la tâche courante
      Générer l'état obtenu à l'aide de la tâche en cours
      Si l'état obtenu n'existe pas **alors**
        Créer l'état obtenu
      Fin si
      Si l'arc n'existe pas **alors**
        Ajouter l'arc de l'état courant à l'état obtenu
      Fin si
      L'état obtenu devient l'état courant
    Fin pour
    Revenir à l'état initial *// Racine de l'ASTD*
  Fin pour

ALGO 3.1 : Algorithme de génération d'un ASTD

ce procédé est répété pour toutes les tâches de toutes les séquences d'enchaînement de l'ensemble Υ.

Pour illustrer cette transformation, l'assemblage du produit $\mathcal{P}$ suivant est traité. $\mathcal{P}$ est constitué des composants A, B, C, D et de leurs tâches respectives associées a, b, c, d et de la tâche u de déchargement du produit fini. L'ensemble Υ des séquences d'enchaînement est par exemple :

$$\begin{aligned} \Upsilon = \{ & \\ & a — c — b — d — u, \\ & a — c — d — b — u, \\ & a — b — c — d — u, \\ & a — b — d — c — u, \\ & a — d — c — b — u, \\ & a — d — b — c — u\} \end{aligned} \tag{3.1}$$

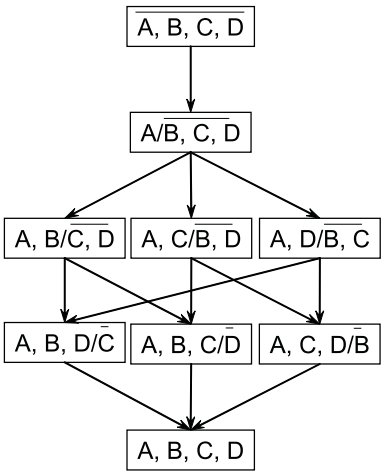


FIG. 3.2 – ASTD représentant l'ensemble Υ des séquences d'enchaînement

À l'aide de l'algorithme 3.1, l'ASTD $\mathcal{A} =< \xi, \tau >$ est obtenu. Il comporte neuf états d'assemblage différents :
$\xi = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, D >, < A, B, \bar{C}, \bar{D} >, < A, \bar{B}, C, \bar{D} >, < A, \bar{B}, C, D >, < A, B, \bar{C}, D >, < A, B, C, \bar{D} >, < A, B, C, D >\}$,
et treize *tâches* différentes :
$\tau = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >, d), (< A, \bar{B}, \bar{C}, D >, b), (< A, \bar{B}, \bar{C}, D >, c), (< A, B, \bar{C}, \bar{D} >, c), (< A, B, \bar{C}, \bar{D} >, d), (< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d), (< A, \bar{B}, C, D >, b), (< A, B, \bar{C}, D >, c), (< A, B, C, \bar{D} >, d)\}$,
plus celle de déchargement $(< A, B, C, D >, u)$ qui n'apparaît pas dans la représentation graphique de l'ASTD, et ce avec seulement cinq tâches d'assemblage (une de chargement du composant de base, trois adjonctions de constituants et une de déchargement). L'ASTD $\mathcal{A}$ peut se représenter comme le montre la figure 3.2.

Avec l'algorithme 3.1, nous pouvons en déduire la propriété 3.1 d'unicité de l'ASTD généré à partir d'un ensemble Υ de séquences d'enchaînement.

PROPRIÉTÉ 3.1 (UNICITÉ DE L'ASTD)
Si Υ est un ensemble de séquences d'enchaînement alors l'ASTD $\mathcal{A}$ généré par l'algorithme 3.1 est unique[6] *et représente de façon biunivoque l'ensemble Υ.*
Qui peut s'écrire :

$$\forall \Upsilon, \exists! \mathcal{A}, \mathcal{M}_{\mathcal{A}}(\Upsilon) = \mathcal{A} | \Upsilon = SEA(\mathcal{A})$$

[6] $\exists!\mathcal{A}$:Il existe de manière unique $\mathcal{A}$

DÉMONSTRATION 3.1
Selon J. VELU *[51], si nous considérons* Υ *comme un langage de Kleene, il lui correspond de façon biunivoque un automate d'état fini* $< \Sigma, \epsilon, I, \mathcal{E}_f, \delta >$, *où* Σ *est l'alphabet de l'automate,* ϵ *est l'ensemble des états de l'automate,* I *est l'état initial,* $\mathcal{E}_f$ *est l'état final,* δ *est la fonction de transition de l'automate définie de* $\epsilon \times \Sigma \rightarrow \epsilon$. *Les états de cet automate sont trivialement les états successifs du produit en cours d'assemblage et les transitions entre les états (adjonction d'un symbole), les tâches. Donc cet automate n'est autre que l'ASTD associé. Nous avons alors* $\forall \Upsilon, \exists ! \mathcal{A}, \mathcal{M}_{\mathcal{A}}(\Upsilon) = \mathcal{A} | \Upsilon = SEA(\mathcal{A})$.

3.3.2 Génération du graphe d'enchaînement

Comme vu ci-avant, le graphe d'enchaînement est le graphe dual d'un ASTD qui représente les mêmes processus d'assemblage, où les nœuds sont les tâches d'assemblage de l'ASTD, et les arcs représentent les enchaînements de ces tâches.

La transformation d'un ASTD en un graphe d'enchaînement est une réécriture de la représentation formelle de l'ASTD en prenant comme nœuds pour le graphe d'enchaînement les tâches entre les états de l'ASTD et à chaque état entre deux tâches de l'ASTD correspond un arc dans le graphe d'enchaînement entre les deux tâches correspondantes.

Un graphe d'enchaînement est un graphe orienté et non simple car il peut exister des paires de nœuds $\{i, j\}$ tel qu'il y ait plusieurs arcs orientés (i, j) et/ou (j, i).

La transformation d'un ASTD en un graphe d'enchaînement est transcrite à l'aide de l'algorithme 3.2. La notion de cocycle[7] est introduite (voir Annexe A), pour résoudre cette phase mathématiquement.

Avec l'exemple de l'ASTD $\mathcal{A}$ de la figure 3.2, les demi-cocycles supérieurs sont :
- $\omega^+_{\mathcal{A}}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}}(< A, \bar{B}, \bar{C}, D >) = \{(< A, \bar{B}, \bar{C}, D >, b), (< A, \bar{B}, \bar{C}, D >, c)\}$
- $\omega^+_{\mathcal{A}}(< A, B, \bar{C}, \bar{D} >) = \{(< A, B, \bar{C}, \bar{D} >, c), (< A, B, \bar{C}, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}}(< A, \bar{B}, C, \bar{D} >) = \{(< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}}(< A, \bar{B}, C, D >) = \{(< A, \bar{B}, C, D >, b)\}$
- $\omega^+_{\mathcal{A}}(< A, B, \bar{C}, D >) = \{(< A, B, \bar{C}, D >, c)\}$

[7] On appelle *cocycle* d'un sommet l'ensemble des arcs qui lui sont incidents.
Le *demi-cocycle supérieur* ω^+ d'un sommet est l'ensemble des arcs sortant de ce sommet.
Le *demi-cocycle inférieur* ω^- d'un sommet est l'ensemble des arcs entrant en ce sommet.

Entrée : $\mathcal{A}$ ASTD

Variable : a, b tâches d'assemblage
$\mathcal{E}$ état initial de a

Sortie : Graphe dual

Pour chaque état de $\mathcal{A}$ **faire**

// Traiter l'état courant
Définir les demi-cocycles supérieurs $\omega^+_{\mathcal{A}}$ de chaque état.

Fin pour

Pour chaque état de $\mathcal{A}$ **faire**

// Traiter l'état courant
Générer toutes les relations de précédence possibles entre les tâches appartenant aux demi-cocycles supérieurs $\omega^+_{\mathcal{A}}$.

Pour chaque relation (a, b) **faire**

Rechercher l'état initial $\mathcal{E}$ de a
Ajouter la relation $(a, b)_{\mathcal{E}}$ dans l'ensemble des demi-cocycles supérieurs $\omega^+_{G_D}$ en a

Fin pour

Fin pour
Avec les demi-cocycles supérieurs $\omega^+_{G_D}$, nous établissons le graphe d'enchaînement G_D correspondant.

ALGO 3.2 : Algorithme de génération du graphe d'enchaînement d'un ASTD

- $\omega^+_{\mathcal{A}}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

Cet algorithme permet de définir les demi-cocycles supérieurs $\omega^+_{\mathcal{A}}$ de chaque état d'un ASTD en recherchant tout état possible suivant, ainsi que la tâche associée. Il permet d'engendrer aussi les demi-cocycles supérieurs $\omega^+_{G_D}$ de chaque nœud du graphe d'enchaînement G_D, afin de le générer.

Avec l'ASTD A de la figure 3.2 et l'algorithme 3.2, les demi-cocycles supérieurs $\omega^+_{G_D}$ suivants du graphe d'enchaînement G_D sont obtenus :

- $\omega^+_{G_D}(a) = \{(a, b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$

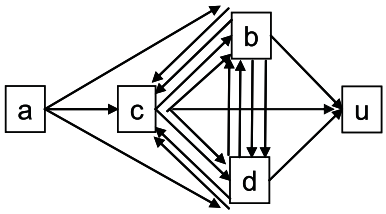


FIG. 3.3 – Graphe d'enchaînement généré

- $\omega^+_{G_D}(b) = \{(b,c)_{<A,\bar{B},\bar{C},D>}, (b,c)_{<A,\bar{B},\bar{C},\bar{D}>}, (b,d)_{<A,\bar{B},\bar{C},\bar{D}>}, (b,d)_{<A,\bar{B},C,\bar{D}>}, (b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_D}(c) = \{(c,b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c,b)_{<A,\bar{B},\bar{C},D>}, (c,d)_{<A,\bar{B},\bar{C},\bar{D}>}, (c,d)_{<A,B,\bar{C},\bar{D}>}, (c,u)_{<A,B,\bar{C},D>}\}$
- $\omega^+_{G_D}(d) = \{(d,b)_{<A,\bar{B},C,\bar{D}>}, (d,b)_{<A,\bar{B},\bar{C},\bar{D}>}\ (d,c)_{<A,\bar{B},\bar{C},\bar{D}>}, (d,c)_{<A,B,\bar{C},\bar{D}>}, (d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_D}(u) = \emptyset$

De ces demi-cocycles supérieurs $\omega^+_{G_D}$, nous en déduisons le graphe d'enchaînement G_D suivant :

$G_D =< E, \sigma, \xi, L >$
$E = \{a, b, c, d, u\}$
$\sigma = \{(a,b),(a,c),(a,d),(b,c),(c,b),(c,d),(d,c),(b,d),(d,b),(b,u),(c,u),(d,u)\}.$
$\xi = \{< \bar{A},\bar{B},\bar{C},\bar{D} >, < A,\bar{B},\bar{C},\bar{D} >, < A,\bar{B},\bar{C},D >, < A,B,\bar{C},\bar{D} >,$
$< A,\bar{B},C,\bar{D} >, < A,\bar{B},C,D >, < A,B,\bar{C},D >, < A,B,C,\bar{D} >\}.$
$L = \{((a,b), < \bar{A},\bar{B},\bar{C},\bar{D} >), ((a,c), < \bar{A},\bar{B},\bar{C},\bar{D} >), ((a,d), < \bar{A},\bar{B},\bar{C},\bar{D} >),$
$((b,c), < A,\bar{B},\bar{C},D >), ((b,c), < A,\bar{B},\bar{C},\bar{D} >), ((c,b), < A,\bar{B},\bar{C},D >),$
$((c,b), < A,\bar{B},\bar{C},\bar{D} >), ((c,d), < A,B,\bar{C},\bar{D} >), ((c,d), < A,\bar{B},\bar{C},\bar{D} >),$
$((d,c), < A,B,\bar{C},\bar{D} >), ((d,c), < A,\bar{B},\bar{C},\bar{D} >), ((b,d), < A,\bar{B},C,\bar{D} >),$
$((b,d), < A,\bar{B},\bar{C},\bar{D} >), ((d,b), < A,\bar{B},C,\bar{D} >), ((d,b), < A,\bar{B},\bar{C},\bar{D} >),$
$((b,u), < A,\bar{B},C,D >), ((c,u), < A,B,\bar{C},D >), ((d,u), < A,B,C,\bar{D} >)\}.$

De là, il est évident que nous avons la propriété d'unicité du graphe d'enchaînement généré suivante :

PROPRIÉTÉ 3.2 (UNICITÉ DU GRAPHE D'ENCHAÎNEMENT D'UN ASTD)
À un ASTD correspond un et un seul graphe d'enchaînement.
Nous pouvons écrire :

$$\forall \mathcal{A}, \exists! G_D, \mathcal{M}_D(\mathcal{A}) = G_D / SEA(\mathcal{A}) = SEA(G_D)$$

Remarque 3.1
La génération des séquences d'enchaînement associées à un graphe d'enchaînement se fait en prenant tous les chemins hamiltoniens *(voir définition A.6 page 128) de ce graphe.*

Remarque 3.2
Pour un ensemble Υ de séquences d'enchaînement, avec les propriétés 3.1 et 3.2, nous avons un et un seul graphe d'enchaînement.

3.3.3 Génération du graphe de précédence

La génération d'un graphe de précédence est composée de deux grandes phases distinctes ; la *simplification du graphe d'enchaînement* et la *détermination du graphe de précédence.*

3.3.3.1 Simplification du graphe d'enchaînement

À partir d'un graphe d'enchaînement, une réduction du nombre d'arcs entre les tâches est effectuée à l'aide d'un ensemble de propriétés. Cette réduction du nombre d'arcs comporte deux cas.

Premier cas : S'il y a *présence*[8] d'arcs entre la tâche α_i et la tâche α_j et pas de *chemin*[9] menant de la tâche α_j à la tâche α_i, alors il est claire que la tâche α_i précède la tâche α_j. Ce cas est exprimé par la propriété 3.3, qui est transcrite dans l'algorithme 3.5.

Propriété 3.3 (Redondance des précédences)
Soit G_D un graphe d'enchaînement, s'il y a au moins un arc de la tâche α_i à la tâche α_j[10]*, et qu'il n'y a pas de chemin de la tâche α_j à la tâche α_i alors la tâche α_i précède la tâche α_j.*
Cela peut s'écrire aussi :

$$\forall G_D, \alpha_i \in G_D, \alpha_j \in G_D, (\alpha_i, \alpha_j) \in \omega^+_{G_D}(\alpha_i), (\alpha_j, \alpha_i) \notin \omega^+_{G_D}(\alpha_j), \nexists Ch(\alpha_j, \alpha_i) \Rightarrow \alpha_i \rightarrow \alpha_j$$

[8] Il existe au moins un arc (α_i, α_j), dans l'ensemble σ du graphe d'enchaînement $G_D =< E, \sigma, \xi, L >$

[9] Un chemin $Ch(\alpha_i, \alpha_j)$ est une succession d'arcs permettant d'aller du nœud α_i au nœud α_j en passant par un nombre fini d'arcs en respectant leur sens.

[10] Il existe au moins un sommet dans le demi-cocycle supérieur $\omega^+_{G_D}(\alpha_i)$

Démonstration 3.2
*Supposons que la tâche α_j précède la tâche α_i dans au moins une séquence d'enchaînement de Υ, alors il existe une séquence x tel que $x = x_{\alpha_j[} — \alpha_j — \alpha_i — x_{]\alpha_i}$. Si une telle séquence existe alors dans le demi-cocycle supérieur $\omega^+_{G_D}(\alpha_j)$, la tâche $(\alpha_j, \alpha_i)_{\mathcal{E}(x_{\alpha_j[})}$ est présente. Il y a contradiction car la condition initiale est «*la tâche $(\alpha_j, \alpha_i)_{\mathcal{E}(x_{\alpha_j[})}$ n'appartient pas au demi-cocycle supérieur de α_j*», alors la tâche α_i précède la tâche α_j.*

Deuxième cas : S'il existe au moins une paire d'arcs dont l'un est de la tâche α_i vers la tâche α_j avec pour état initial $\mathcal{E}$, et l'autre est de la tâche α_j à la tâche α_i avec pour état initial $\mathcal{E}$ c'est à dire $x_1 \equiv x_{1_{\alpha_i[}} — \alpha_i — \alpha_j — x_{1_{]\alpha_j}}$ et $x_2 \equiv x_{2_{\alpha_j[}} — \alpha_j — \alpha_i — x_{2_{]\alpha_i}}$ où $x_{1_{\alpha_i[}} = x_{2_{\alpha_j[}}$ et $x_{1_{]\alpha_j}} = x_{2_{]\alpha_i}}$, alors toutes les paires d'arcs vérifiant cette hypothèse sont supprimées. Après ces simplifications, deux situations s'offrent à nous, pour les deux tâches, soit tous les arcs sont supprimés, alors les deux tâches sont aussi dites indifférentes, soit il reste des arcs entre les deux sommets après simplification, alors les deux tâches sont dites en relation de *précédence conditionnelle* (notée $\alpha_i P c \alpha_j$).

Définition 3.5 (Indifférence dans les graphes de précédence [11])
Dans un graphe de précédence, deux tâches α_i et α_j sont dites indifférentes s'il n'existe pas d'arc ou de chemin entre eux.

Avec la définition 3.5, il est possible d'écrire la définition suivante :

Définition 3.6 (Indifférence dans un graphe d'enchaînement)
Dans un graphe d'enchaînement, deux tâches α_i et α_j sont dites indifférentes s'il n'existe pas d'arc ou de chemin entre eux ou si pour chaque enchaînement (α_i, α_j) d'état initial $\mathcal{E}$, il existe un enchaînement (α_j, α_i) d'état initial $\mathcal{E}$.

Propriété 3.4 (Simplification)
Soit G_D un graphe d'enchaînement, s'il y a au moins un arc de la tâche α_i à la tâche α_j avec pour état initial α , et au moins un arc de la tâche α_j à la tâche α_i avec pour état initial $\mathcal{E}$, alors cela traduit l'existence de deux séquences x_1 et x_2 telle que $x = x_{1_{\alpha_i[}} — \alpha_i — \alpha_j — x_{1_{]\alpha_j}}$ et $x_2 = x_{2_{\alpha_j[}} — \alpha_j — \alpha_i — x_{2_{]\alpha_i}}$, les tâches α_i et α_j sont alors dites indifférentes. Dans ce cas, nous pouvons supprimer la paire d'arcs.
Cette propriété peut être notée :

Entrée : Graphe dual, sommet initial, sommet final

Sortie : Nombre d'arcs entre deux sommets

Pour l'ensemble ω^+ des demi-cocycles supérieurs du graphes dual **faire**

// Recherche de l'ensemble des successeurs du sommet initial

Compter le nombre d'occurrences de l'arc du sommet initial au sommet final dans ω_G^+(sommet initial)

Fin pour

ALGO 3.3 : Algorithme de calcul du nombre d'arcs orientés entre deux sommets du graphe d'enchaînement

$$\forall G_D, \alpha_i, \alpha_j \in G_D, (\alpha_i, \alpha_j)_{\mathcal{E}} \in \omega^+_{G_D}(\alpha_i), (\alpha_j, \alpha_i)_{\mathcal{E}} \in \omega^+_{G_D}(\alpha_j) \Rightarrow$$
$$\exists x_1, \exists x_2 / x_1 = x_{1_{\alpha_i[}} \text{ — } \alpha_i \text{ — } \alpha_j \text{ — } x_{1_{]\alpha_j}}, x_2 = x_{2_{\alpha_j}} \text{ — } \alpha_j \text{ — } \alpha_i \text{ — } x_{2_{\alpha_i}},$$
$$\mathcal{E}(x_{1_{\alpha_i[}}) = \mathcal{E}(x_{2_{\alpha_j[}}), \mathcal{E}(x_{1_{]\alpha_j}}) = \mathcal{E}(x_{2_{]\alpha_i}}) \Rightarrow$$
$$\alpha_i \equiv_{\{\alpha_i[\}} \alpha_j$$

DÉMONSTRATION 3.3

Supposons que l'état engendré par la séquence $x_{1_{\alpha_i[}}$ soit différent de l'état engendré par la séquence $x_{2_{\alpha_j[}}$ alors l'état initial $\alpha_j[$ de l'arc (α_j, α_i) est différent de l'état initial $\alpha_i[$ de l'arc (α_i, α_j). Il y a contradiction car il est supposé que l'état initial $\alpha_i[$ de la séquence $x_{1_{\alpha_i[}}$ est le même que l'état initial $\alpha_j[$ de la séquence $x_{2_{\alpha_j[}}$, alors $x_{1_{\alpha_i[}} = x_{2_{\alpha_j[}}$.
Comme $\mathcal{E}(x_{1_{\alpha_i[}}) = \mathcal{E}(x_{2_{\alpha_j[}})$, l'état engendré $\mathcal{E}(x_{1_{\alpha_i[}} \text{ — } \alpha_i \text{ — } \alpha_j)$ est identique à l'état engendré $\mathcal{E}(x_{2_{\alpha_j[}} \text{ — } \alpha_i \text{ — } \alpha_j)$ alors $\mathcal{E}(x_{1_{]\alpha_j}}) = \mathcal{E}(x_{2_{]\alpha_i}})$ car $\mathcal{E}(x_1) = \mathcal{E}(x_2)$.
Comme $(\alpha_i, \alpha_j)_{\mathcal{E}} \in \omega^+_{G_D}(\alpha_i)$ il existe une séquence x_1 telle que $x_1 = x_{1_{\alpha_i[}} \text{ — } \alpha_i \text{ — } \alpha_j \text{ — } x_{1_{]\alpha_j}}$. Nous procédons de même pour $(\alpha_j, \alpha_i)_{\mathcal{E}} \in \omega^+_{G_D}(\alpha_j)$. La première implication est alors démontrée.
Pour la deuxième implication, il est évident que si deux séquences d'enchaînement de ce type existent alors il y a précédence de la tâche α_i sur la tâche α_j et précédence de la tâche α_j sur la tâche α_i, dans ce cas nous disons qu'elles sont indifférentes.

Pour la deuxième de ces situations, qui est, nous le rappelons, l'existence d'arcs entre les deux tâches d'état initial différents pour chacune des paires d'arcs. S'il existe un nombre $n_{(\alpha_i,\alpha_j)}$ d'arcs (α_i, α_j) et un nombre $n_{(\alpha_j,\alpha_i)}$ d'arcs (α_j, α_i)[11] non nuls ne

[11] $n_{(\alpha_i,\alpha_j)}$ peut être différent de $n_{(\alpha_j,\alpha_i)}$.

Entrée : Graphe dual

Variable : sommet courant
sommet initial

Sortie : Nombre d'arc entre chaque paire orientée de sommets

Pour chaque sommet initial **faire**

// Compter le nombre d'arcs partant de ce sommet pour chaque sommet destination

Pour chaque sommet de l'ensemble des demi-cocycles supérieurs (ω^+) **faire**

// Compter le nombre d'occurrence d'un sommet
Appel de l'algorithme 3.3 de calcul du nombre d'arcs orientés entre deux sommets du graphe d'enchaînement avec le sommet initial et le sommet courant

Fin pour

Fin pour

ALGO 3.4 : Algorithme de calcul du nombre d'arcs entre chaque paire orientée de sommets d'un graphe d'enchaînement

pouvant pas être réduits avec les propriétés 3.3 et 3.4, alors le graphe d'enchaînement simplifié est appelé un *graphe d'enchaînement complexe*. Si les différentes simplifications nous conduisent à un graphe d'enchaînement complexe, un hypergraphe de précédence sera alors engendré. Cette génération d'hypergraphe de précédence sera développée dans la section 3.5.2 p. 80.

Après ces simplifications successives, un graphe de précédence est généré si le graphe d'enchaînement n'est pas un graphe d'enchaînement complexe.

Il est nécessaire d'introduire la notion de *nombre d'arcs* (voir définition A.9 page 130) afin de simplifier ce graphe mathématiquement. Le nombre d'arcs m pour chaque paire de sommets a et b d'un graphe G est calculé à l'aide des algorithmes 3.3 et 3.4 et est noté $m_G(\alpha_i, \alpha_j)$. Ce nombre d'arcs est le nombre d'occurrences du sommet b dans l'ensemble des successeurs du sommet α_i.

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement G_D avant simplification est :

- $m_{G_D}(a, b) = 1$

- $m_{G_D}(a, c) = 1$
- $m_{G_D}(a, d) = 1$
- $m_{G_D}(b, c) = 2$
- $m_{G_D}(b, d) = 2$
- $m_{G_D}(b, u) = 1$
- $m_{G_D}(c, b) = 2$
- $m_{G_D}(c, d) = 2$
- $m_{G_D}(c, u) = 1$
- $m_{G_D}(d, b) = 2$
- $m_{G_D}(d, c) = 2$
- $m_{G_D}(d, u) = 1$

De là il est facile de remarquer que le graphe d'enchaînement n'est pas un graphe simple, car il possède plusieurs arcs entre certains sommets, comme par exemple entre les sommets b et c.

À l'aide de l'algorithme 3.5 de simplification des graphes d'enchaînement, le nombre d'arcs est réduit entre chaque paire de sommets en suivant le procédé précédemment cité. Pour chacune des paires de sommets, s'il n'existe pas d'arcs avant simplification, les deux sommets sont dits indépendants, sinon si après simplification, il n'y a plus d'arcs entre deux sommets, ceux-ci sont dits indifférents. S'il existe une paire de sommets comportant des arcs dans un sens et pas dans l'autre, alors l'enchaînement des tâches suit ces arcs, en revanche s'il existe des arcs dans les deux sens entre ces sommets, un procédé de simplification est mis en place. Pour simplifier deux arcs opposés entre deux sommets, il est nécessaire d'avoir pour chacun d'eux le même état initial[12]. Cette procédure est répétée jusqu'à la fin du traitement de toutes les paires d'arcs de tous les couples de sommets.

L'algorithme, avec l'exemple précédent, établi les relations suivantes entre chaque paire de tâches déjà en relation dans le graphe d'enchaînement :

- la tâche a précède la tâche b ;
- la tâche a précède la tâche c ;
- la tâche a précède la tâche d ;
- la tâche b est indifférente de la tâche c, car :
  - la tâche b est indifférente de la tâche c par rapport à l'état $< A, \bar{B}, \bar{C}, \bar{D} >$;
  - la tâche b est indifférente de la tâche c par rapport à l'état $< A, \bar{B}, \bar{C}, D >$;
- la tâche b est indifférente de la tâche d, car :

[12] Nous appelons état initial le nœud initial de la tâche de départ dans l'ASTD correspondant

Entrée : Graphe dual

Variable : x, y sommets

$\mathcal{E}$ état

Sortie : Graphe dual simplifié

Pour chaque sommet x **faire**

// Rechercher les enchaînements opposés de deux tâches en partant du même état

Si $m_{G_D}(x,y) = 0$ **alors**

//Il n'existe pas d'arcs (x,y)

Si $m_{G_D}(y,x) = 0$ **alors**

$x \mathcal{I} y$

Sinon

$y \rightarrow x$

Fin si

Sinon

// Il existe au moins un arc (x,y)

Si $m_{G_D}(y,x) = 0$ **alors**

$x \rightarrow y$

Sinon

Si $(x,y)_{\mathcal{E}} \in \omega^{+}_{G_D}(x) \wedge (y,x)_{\mathcal{E}} \in \omega^{-}_{G_D}(x)$ **alors**

Les arcs $(x,y)_{\mathcal{E}}$ et $(y,x)_{\mathcal{E}}$ sont supprimés et l'algorithme est relancé sur le même sommet.

Fin si

Fin si

Fin si

Fin pour

ALGO 3.5 : Algorithme de simplification du graphe d'enchaînement

- la tâche b est indifférente de la tâche d par rapport à l'état $< A, \bar{B}, \bar{C}, \bar{D} >$;
- la tâche b est indifférente de la tâche d par rapport à l'état $< A, \bar{B}, C, \bar{D} >$;

– la tâche c est indifférente de la tâche d, car :

- la tâche c est indifférente de la tâche d par rapport à l'état $< A, \bar{B}, \bar{C}, \bar{D} >$;
- la tâche c est indifférente de la tâche d par rapport à l'état $< A, B, \bar{C}, \bar{D} >$;

– la tâche b précède la tâche u ;
– la tâche c précède la tâche u ;
– la tâche d précède la tâche u.

De là, le graphe d'enchaînement simplifié suivant est généré :
$G_{D_s} =< E_s, \sigma_s, \xi, L >$ où
$E_s = \{a, b, c, d, u\}$,
$\sigma_s = \{(a, b), (a, c), (a, d), (b, u), (c, u), (d, u)\}$,
$\xi_s = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, C, D >, < A, B, \bar{C}, D >, < A, B, C, \bar{D} >\}$
et $L_s = \{((a, b), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a, c), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a, d), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >),$
$((b, u), < A, \bar{B}, C, D >), ((c, u), < A, B, \bar{C}, D >), ((d, u), < A, B, C, \bar{D} >)\}$

Après la simplification du graphe d'enchaînement, nous passons à la phase de détermination du graphe de précédence.

3.3.3.2 Détermination du graphe de précédence

Comme vu ci-avant, si le graphe d'enchaînement simplifié n'est pas un graphe d'enchaînement complexe alors il est équivalent au graphe de précédence associé à l'ensemble des séquences d'enchaînement.

Le graphe de précédence Gp est engendré, en affectant respectivement les nœuds et les arcs du graphe d'enchaînement simplifié aux nœuds et aux arcs du graphe de précédence. Cette méthode donne le graphe de précédence G_P suivant :
$G_P =< E, U >$ où $E = \{a, b, c, d, u\}$ et $U = \{(a, b), (a, c), (a, d), (b, u), (c, u), (d, u)\}$

Le graphe de précédence ainsi généré peut être représenté par la figure 3.4.

Il est évident que le graphe ainsi engendré est unique, nous avons donc la propriété d'unicité du graphe de précédence généré suivante.

Propriété 3.5 (Unicité du graphe de précédence généré)
Soit G_D un graphe d'enchaînement, et G son graphe de précédence engendré par notre méthode, nous avons $SEA(G_D) = SEA(G)$ si et seulement si G_D peut être représenté par un seul graphe de précédence.
Soit :

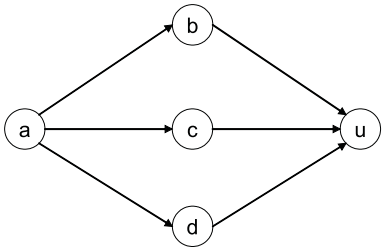


FIG. 3.4 – Graphe de précédence généré

$$\forall G_D, \exists! G / SEA(G_D) = SEA(G) \Rightarrow$$
$$\exists! G', \mathcal{M}_G(G_D) = G' / SEA(G_D) = SEA(G') \; et \; G' = G$$

3.4 Exemple

Afin d'illustrer cette méthode, un ensemble de cas différents d'un exemple est développé ci-après. Soit un produit $\mathcal{P}$ constitué de quatre composants A, B, C, D, ces composants sont « assemblés » par leurs tâches respectives a, b, c, d. Pour cet exemple, un ensemble de cas possibles (Cas i) est étudié, où les ensembles Υ_i des séquences admissibles seront différents dans chacun d'entre eux.

3.4.1 Cas 1 :

L'ensemble Υ_1 des séquences d'enchaînement est :

$$\Upsilon_1 = \{\ a \text{ — } b \text{ — } c \text{ — } d \text{ — } u\}$$

Avec l'ensemble Υ_1, l'ASTD $\mathcal{A}_1 =< \xi_1, \tau_1 >$ de la figure 3.5 est obtenu :
$\xi_1 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, B, \bar{C}, \bar{D} >, < A, B, C, \bar{D} >, < A, B, C, D >$
$\}$
et $\tau_1 = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, B, \bar{C}, \bar{D} >, c), (< A, B, C, \bar{D} >$
$, d), (< A, B, C, D >, u)\}$

Les demi-cocycles supérieurs de $\mathcal{A}_1$ sont :
- $\omega^+_{\mathcal{A}_1}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}_1}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, b)\}$
- $\omega^+_{\mathcal{A}_1}(< A, B, \bar{C}, \bar{D} >) = \{(< A, B, \bar{C}, \bar{D} >, c)\}$

- $\omega^+_{\mathcal{A}_1}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_1}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

La méthode de transformation d'un ASTD en un le graphe d'enchaînement donne le graphe G_{D_1} représenté par la figure 3.5(b) avec $G_{D_1} = (E_1, \sigma_1, \xi_1, L_1)$ où
$E_1 = \{a, b, c, d, u\}$ et
$\sigma_1 = \{(a, b), (b, c), (c, d), (d, u)\}$,
$\xi_1 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, B, \bar{C}, \bar{D} >, < A, B, C, \bar{D} >, < A, B, C, D >$ } et
$L_1 = \{((a, b), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((b, c), < A, \bar{B}, \bar{C}, \bar{D} >), ((c, d), < A, B, \bar{C}, \bar{D} >)$,
$((d, u), < A, B, C, \bar{D} >)\}$
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_1}}(a) = \{(a, b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_1}}(b) = \{(b, c)_{<A,\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_1}}(c) = \{(c, d)_{<A,B,\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_1}}(d) = \{(d, u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_1}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement est :

- $m_{G_{D_1}}(a, b) = 1$
- $m_{G_{D_1}}(b, c) = 1$
- $m_{G_{D_1}}(c, d) = 1$
- $m_{G_{D_1}}(d, u) = 1$

Dans ce cas très particulier, la méthode de simplification du nombre d'arcs n'influe pas sur celui-ci, car le graphe d'enchaînement est linéaire. D'après la propriété 3.3, un graphe d'enchaînement linéaire ne peut pas avoir d'arc élagué.

Nous voyons dans ce cas très simple, à l'aide de la figure 3.5(c), que le graphe engendré par notre méthode, est bien le graphe de précédence associé (ici trivialement) à l'ensemble des séquences d'enchaînement.

3.4.2 Cas 2 :

L'ensemble Υ_2 des séquences d'enchaînement est :

$$\Upsilon_2 = \{$$
$$a — c — b — d — u,$$
$$a — c — d — b — u\}$$

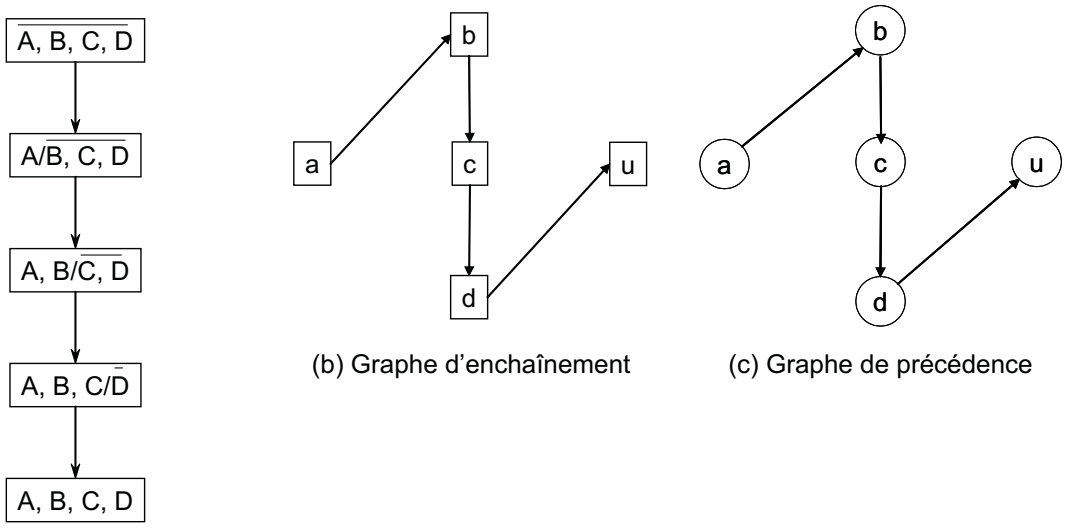


FIG. 3.5 – Cas 1

Avec l'ensemble Υ_2, l'ASTD $\mathcal{A}_2 =< \xi_2, \tau_2 >$ de la figure 3.6 est obtenu :
$\xi_2 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, B, \bar{C}, \bar{D} >, < A, B, C, \bar{D} >, < A, \bar{B}, C, D >$
$, < A, B, C, D >\}$
et $\tau_2 = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, B, \bar{C}, \bar{D} >, c), (< A, B, C, \bar{D} >$
$, d), (< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, D >, b), (< A, B, C, D >, u)\}$

les demi-cocycles supérieurs de l'ASTD $\mathcal{A}_2$ sont :
- $\omega^+_{\mathcal{A}_2}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}_2}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, c)\}$
- $\omega^+_{\mathcal{A}_2}(< A, \bar{B}, C, \bar{D} >) = \{(< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_2}(< A, \bar{B}, C, D >) = \{(< A, \bar{B}, C, D >, b)\}$
- $\omega^+_{\mathcal{A}_2}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_2}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

La méthode de transformation d'un ASTD en un graphe d'enchaînement donne le graphe G_{D_2} représenté par la figure 3.6(b) avec $G_{D_2} = (E_2, \sigma_2, \xi_2, L_2)$ où
$E_2 = \{a, b, c, d, u\}$ et
$\sigma_2 = \{(a, c), (b, d), (b, u), (c, b), (c, d), (d, b), (d, u)\}$,
$\xi_2 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, B, \bar{C}, \bar{D} >, < A, B, C, \bar{D} >, < A, \bar{B}, C, D >$
$, < A, B, C, D >\}$
et $L_2 = \{((a, c), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((b, d), < A, \bar{B}, C, \bar{D} >), ((b, u), < A, \bar{B}, C, D >),$

$((c,b), <A,\bar{B},\bar{C},\bar{D}>)$, $((c,d), <A,\bar{B},\bar{C},\bar{D}>)$, $((d,b), <A,\bar{B},C,\bar{D}>)$,
$((d,u), <A,B,C,\bar{D}>)\}$,
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_2}}(a) = \{(a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_2}}(b) = \{(b,d)_{<A,\bar{B},C,\bar{D}>}, (b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_2}}(c) = \{(c,b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c,d)_{<A,\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_2}}(d) = \{(d,b)_{<A,\bar{B},C,\bar{D}>}, (d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_2}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement est :

- $m_{G_{D_2}}(a,c) = 1$
- $m_{G_{D_2}}(b,d) = 1$
- $m_{G_{D_2}}(b,u) = 1$
- $m_{G_{D_2}}(c,b) = 1$
- $m_{G_{D_2}}(c,d) = 1$
- $m_{G_{D_2}}(d,b) = 1$
- $m_{G_{D_2}}(d,u) = 1$

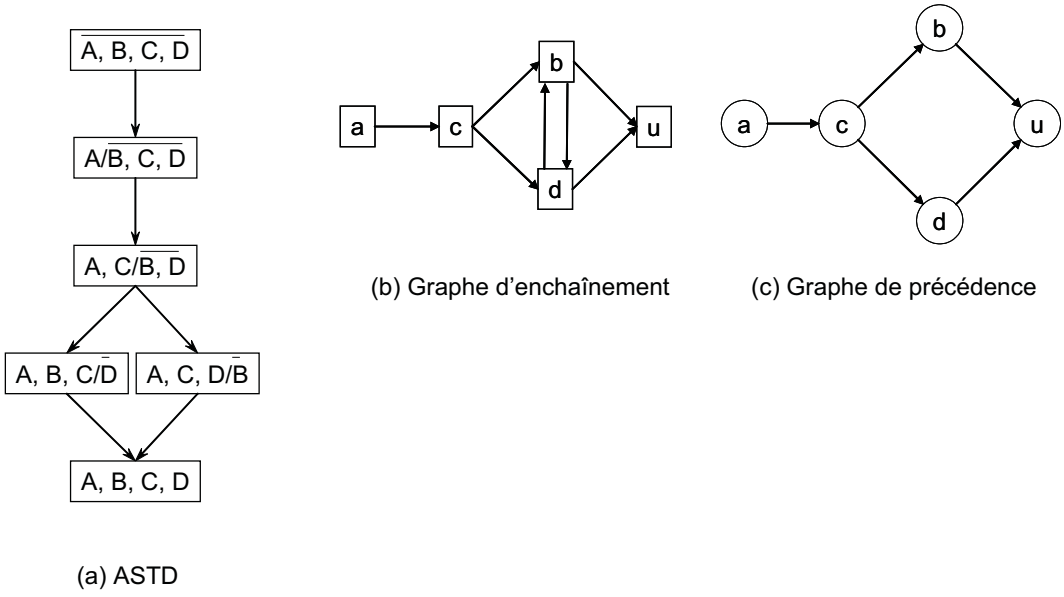
L'algorithme permet de simplifier la paire d'arcs entre les tâches b et d, ce qui donne l'indifférence entre ces deux tâches. L'ensemble des autres enchaînements de tâches reste inchangé de par la propriété 3.3.

Le graphe d'enchaînement simplifié $G_{D_{2s}}$ est alors : $G_{D_{2s}} = (E_{2s}, \sigma_{2s}, \xi_{2s}, L_{2s})$ où
$E_{2s} = \{a,b,c,d,u\}$,
$\sigma_{2s} = \{(a,c), (b,u), (c,b), (c,d), (d,u)\}$,
$\xi_{2s} = \{<\bar{A},\bar{B},\bar{C},\bar{D}>, <A,\bar{B},\bar{C},\bar{D}>, <A,B,\bar{C},\bar{D}>, <A,B,C,\bar{D}>, <A,\bar{B},C,D>$
$, <A,B,C,D>\}$
et $L_{2s} = \{((a,c), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((b,u), <A,\bar{B},C,D>), ((c,b), <A,\bar{B},\bar{C},\bar{D}>)$,
$((c,d), <A,\bar{B},\bar{C},\bar{D}>), ((d,u), <A,B,C,\bar{D}>)\}$,
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_{2s}}}(a) = \{(a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{2s}}}(b) = \{(b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_{2s}}}(c) = \{(c,b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c,d)_{<A,\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{2s}}}(d) = \{(d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_{2s}}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement simplifié est :

- $m_{G_{D_{2s}}}(a,c) = 1$
- $m_{G_{D_{2s}}}(b,u) = 1$



(a) ASTD

(b) Graphe d'enchaînement

(c) Graphe de précédence

FIG. 3.6 – Cas 2

- $m_{G_{D_{2s}}}(c, b) = 1$
- $m_{G_{D_{2s}}}(c, d) = 1$
- $m_{G_{D_{2s}}}(d, u) = 1$

Cet exemple illustre un cas de simplification d'une paire d'arcs opposés de même état initial où deux tâches sont indifférentes.

À l'aide de la figure 3.6(c), nous voyons que le graphe obtenu $G_{D_{2s}}$ est bien le graphe de précédence associé à l'ensemble Υ_2 des séquences d'enchaînement, il est exhaustif et valide.

3.4.3 Cas 3 :

L'ensemble Υ_3 des séquences d'enchaînement est :

$$\begin{gathered}\Upsilon_3 = \{ \\ a — c — b — d — u, \\ a — d — b — c — u\}\end{gathered}$$

Avec l'ensemble Υ_3, l'ASTD $\mathcal{A}_3 =< \xi_3, \tau_3 >$ de la figure 3.7 est obtenu :
$\xi_3 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, D >, < A, B, \bar{C}, \bar{D} >, < A, \bar{B}, C, D >$
$, < A, B, C, \bar{D} >, < A, B, C, D >\}$

et $\tau_3 = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, B, \bar{C}, \bar{D} >, c), (< A, B, C, \bar{D} >, d), (< A, \bar{B}, \bar{C}, \bar{D} >, d), (< A, \bar{B}, \bar{C}, D >, c), (< A, \bar{B}, C, D >, b), (< A, B, C, D >, u)\}$

Les demi-cocycles supérieurs de l'ASTD $\mathcal{A}_3$ sont :

- $\omega^+_{\mathcal{A}_3}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}_3}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, \bar{B}, \bar{C}, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_3}(< A, \bar{B}, \bar{C}, D >) = \{(< A, \bar{B}, \bar{C}, D >, c)\}$
- $\omega^+_{\mathcal{A}_3}(< A, B, \bar{C}, \bar{D} >) = \{(< A, B, \bar{C}, \bar{D} >, c)\}$
- $\omega^+_{\mathcal{A}_3}(< A, \bar{B}, C, D >) = \{(< A, \bar{B}, C, D >, b)\}$
- $\omega^+_{\mathcal{A}_3}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_3}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

La méthode de transformation d'un ASTD en un graphe d'enchaînement donne le graphe G_{D_3} représenté par la figure 3.7(b) avec $G_{D_3} = (E_3, \sigma_3, \xi_3, L_3)$ où
$E_3 = \{a, b, c, d, u\}$,
$\sigma_3 = \{(a, b), (a, c), (b, c), (b, u), (c, b), (c, d), (d, c), (d, u)\}$,
$\xi_3 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, D >, < A, B, \bar{C}, \bar{D} >, < A, \bar{B}, C, D >, < A, B, C, \bar{D} >, < A, B, C, D >\}$
et $L = \{((a, b), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a, c), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((b, c), < A, \bar{B}, \bar{C}, \bar{D} >), ((b, u), < A, \bar{B}, C, D >), ((c, b), < A, \bar{B}, \bar{C}, D >), ((c, d), < A, B, \bar{C}, \bar{D} >), ((d, c), < A, \bar{B}, \bar{C}, \bar{D} >), ((d, u), < A, B, C, \bar{D} >)\}$
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_3}}(a) = \{(a, b)_{<\bar{A}, \bar{B}, \bar{C}, \bar{D}>}, (a, c)_{<\bar{A}, \bar{B}, \bar{C}, \bar{D}>}\}$
- $\omega^+_{G_{D_3}}(b) = \{(b, c)_{<A, \bar{B}, \bar{C}, \bar{D}>}, (b, u)_{<A, \bar{B}, C, D>}\}$
- $\omega^+_{G_{D_3}}(c) = \{(c, b)_{<A, \bar{B}, \bar{C}, D>}, (c, d)_{<A, B, \bar{C}, \bar{D}>}\}$
- $\omega^+_{G_{D_3}}(d) = \{(d, c)_{<A, \bar{B}, \bar{C}, \bar{D}>}, (d, u)_{<A, B, C, \bar{D}>}\}$
- $\omega^+_{G_{D_3}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement est :

- $m_{G_{D_3}}(a, b) = 1$
- $m_{G_{D_3}}(a, d) = 1$
- $m_{G_{D_3}}(b, c) = 1$
- $m_{G_{D_3}}(b, u) = 1$
- $m_{G_{D_3}}(c, b) = 1$
- $m_{G_{D_3}}(c, d) = 1$
- $m_{G_{D_3}}(d, c) = 1$
- $m_{G_{D_3}}(d, u) = 1$

Dans ce cas, l'algorithme de simplification n'influe pas sur le graphe d'enchaînement car aucune paire d'arcs ne respecte les conditions de simplification. Les paires d'arcs

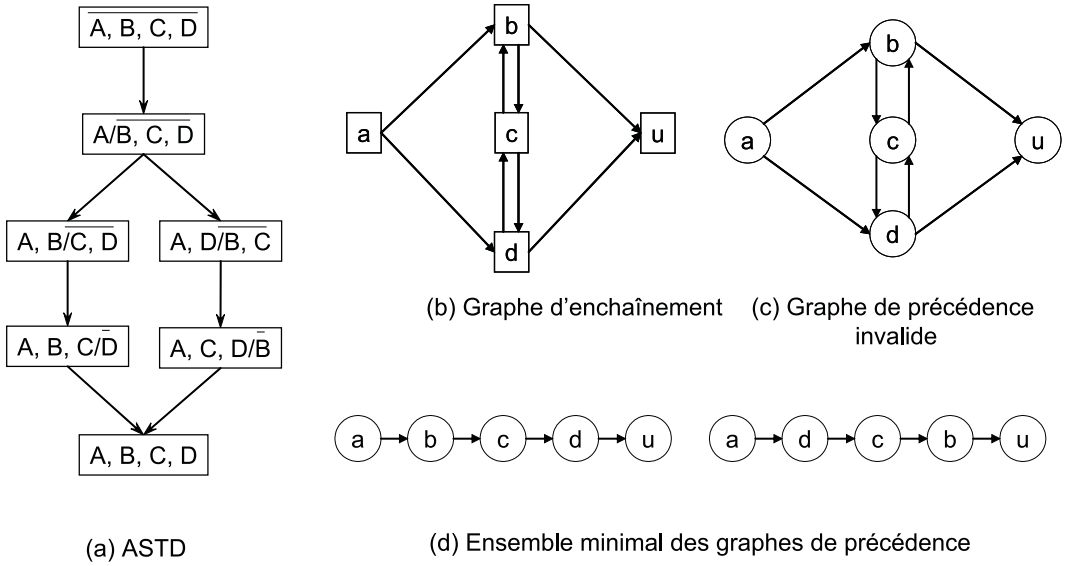


FIG. 3.7 – Cas 3

opposés sont, par exemple entre les tâches b et c d'état initial $< A, \bar{B}, \bar{C}, \bar{D} >$ pour l'arc allant de la tâche b à la tâche c et d'état initial $< A, \bar{B}, \bar{C}, D >$ pour l'arc allant de c à b.

Pour la figure 3.7, le graphe généré, avec la méthode de génération des *graphes de précédence*, n'est pas un graphe de précédence valide car il existe des circuits dans ce graphe.

3.4.4 Cas 4 :

L'ensemble Υ_4 des séquences d'enchaînement est :

$$\begin{gathered}\Upsilon_4 = \{ \\ a — c — b — d — u, \\ a — c — d — b — u, \\ a — b — c — d — u, \\ a — d — c — b — u\}\end{gathered}$$

Avec l'ensemble Υ_4, l'ASTD $\mathcal{A}_4 =< \xi_4, \tau_4 >$ de la figure 3.8 est obtenu :
$\xi_4 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, D >, < A, B, \bar{C}, \bar{D} >, < A, \bar{B}, C, \bar{D} >$
$, < A, \bar{B}, C, D >, < A, B, C, \bar{D} >, < A, B, C, D >\}$
et $\tau_4 = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >$

$, d), (< A, \bar{B}, \bar{C}, D >, c), (< A, B, \bar{C}, \bar{D} >, c), (< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d), (< A, \bar{B}, C, D >, b), (< A, B, C, \bar{D} >, d), (< A, B, C, D >, u)\}$

Les demi-cocycles supérieurs de l'ASTD $\mathcal{A}_4$ sont :

- $\omega^+_{\mathcal{A}_4}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}_4}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, b), (< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_4}(< A, \bar{B}, \bar{C}, D >) = \{(< A, \bar{B}, \bar{C}, D >, c)\}$
- $\omega^+_{\mathcal{A}_4}(< A, B, \bar{C}, \bar{D} >) = \{(< A, B, \bar{C}, \bar{D} >, c)\}$
- $\omega^+_{\mathcal{A}_4}(< A, \bar{B}, C, \bar{D} >) = \{(< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_4}(< A, \bar{B}, C, D >) = \{(< A, \bar{B}, C, D >, b)\}$
- $\omega^+_{\mathcal{A}_4}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_4}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

La méthode de transformation d'un ASTD en un graphe d'enchaînement donne le graphe G_{D_4} représenté par la figure 3.8(b) avec $G_{D_4} = (E_4, \sigma_4, \xi_4, L_4)$ où
$E_4 = \{a, b, c, d, u\}$ et
$\sigma_4 = \{(a, b), (a, c), (a, d), (b, c), (b, d), (b, u), (c, b), (c, b), (c, d), (c, d), (d, b), (d, c), (d, u)\}$,
$\xi_4 = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, D >, < A, B, \bar{C}, \bar{D} >, < A, \bar{B}, C, \bar{D} >, < A, \bar{B}, C, D >, < A, B, C, \bar{D} >, < A, B, C, D >\}$
et $L_4 = \{((a, b), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a, c), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a, d), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >),$
$((b, c), < A, \bar{B}, \bar{C}, \bar{D} >), ((b, d), < A, \bar{B}, C, \bar{D} >), ((b, u), < A, \bar{B}, C, D >),$
$((c, b), < A, \bar{B}, \bar{C}, D >), ((c, b), < A, \bar{B}, \bar{C}, \bar{D} >), ((c, d), < A, \bar{B}, \bar{C}, \bar{D} >),$
$((c, d), < A, B, \bar{C}, \bar{D} >), ((d, b), < A, \bar{B}, C, \bar{D} >),$
$((d, c), < A, \bar{B}, \bar{C}, \bar{D} >), (d, u), < A, B, C, \bar{D} >)\}$
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_4}}(a) = \{(a, b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_4}}(b) = \{(b, c)_{<A,\bar{B},\bar{C},\bar{D}>}, (b, d)_{<A,\bar{B},C,\bar{D}>}, (b, u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_4}}(c) = \{(c, b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c, b)_{<A,\bar{B},\bar{C},D>}, (c, d)_{<A,\bar{B},\bar{C},\bar{D}>}, (c, d)_{<A,B,\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_4}}(d) = \{(d, b)_{<A,\bar{B},C,\bar{D}>}, (d, c)_{<A,\bar{B},\bar{C},\bar{D}>}, (d, u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_4}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement est :

- $m_{G_{D_4}}(a, b) = 1$
- $m_{G_{D_4}}(a, c) = 1$
- $m_{G_{D_4}}(a, d) = 1$
- $m_{G_{D_4}}(b, c) = 1$
- $m_{G_{D_4}}(b, d) = 1$
- $m_{G_{D_4}}(b, u) = 1$

- $m_{G_{D_4}}(c,b) = 2$
- $m_{G_{D_4}}(c,d) = 2$
- $m_{G_{D_4}}(d,b) = 1$
- $m_{G_{D_4}}(d,c) = 1$
- $m_{G_{D_4}}(d,u) = 1$

L'algorithme de simplification permet de supprimer les arcs entre les tâches b et d, afin de les rendre *indépendantes* et de supprimer une paire d'arcs entre les tâches b et c ainsi qu'entre les tâches c et d afin de les rendre *indifférentes* ($b \equiv_{<A,\bar{B},\bar{C},\bar{D}>} c$ et $c \equiv_{<A,\bar{B},\bar{C},\bar{D}>} d$). Comme la tâche c est indifférente des tâches b et d d'une part et que d'autre part la tâche c précède les tâches b et d, il est possible avec la propriété 3.3 et 3.4 de dire que la tâche c précède les tâches b et d.

Le graphe d'enchaînement G_{D_4} devient le graphe d'enchaînement simplifié $G_{D_{4s}}$:
$G_{D_{4s}} = (E_{4s}, \sigma_{4s}, \xi_{4s}, L_{4s})$ où
$E_{4s} = \{a, b, c, d, u\}$,
$\sigma_{4s} = \{(a,b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (b,u)_{<A,\bar{B},C,D>}, (c,b)_{<A,\bar{B},\bar{C},D>}, (c,d)_{<A,B,\bar{C},\bar{D}>}, (d,u)_{<A,B,C,\bar{D}>}\}$,
$\xi_{4s} = \{<\bar{A},\bar{B},\bar{C},\bar{D}>, <A,\bar{B},\bar{C},\bar{D}>, <A,\bar{B},\bar{C},D>, <A,B,\bar{C},\bar{D}>, <A,\bar{B},C,\bar{D}>, <A,\bar{B},C,D>, <A,B,C,\bar{D}>, <A,B,C,D>\}$
et $L_{4s} = \{((a,b), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((a,c), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((a,d), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((b,u), <A,\bar{B},C,D>), ((c,b), <A,\bar{B},\bar{C},D>), ((c,d), <A,B,\bar{C},\bar{D}>), ((d,u), <A,B,C,\bar{D}>)\}$
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_{4s}}}(a) = \{(a,b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(b) = \{(b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_{4s}}}(c) = \{(c,b)_{<A,\bar{B},\bar{C},D>}, (c,d)_{<A,B,\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(d) = \{(d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement simplifié est :

- $m_{G_{D_{4s}}}(a,b) = 1$
- $m_{G_{D_{4s}}}(a,c) = 1$
- $m_{G_{D_{4s}}}(a,d) = 1$
- $m_{G_{D_{4s}}}(b,u) = 1$
- $m_{G_{D_{4s}}}(c,b) = 1$
- $m_{G_{D_{4s}}}(c,d) = 1$
- $m_{G_{D_{4s}}}(d,u) = 1$

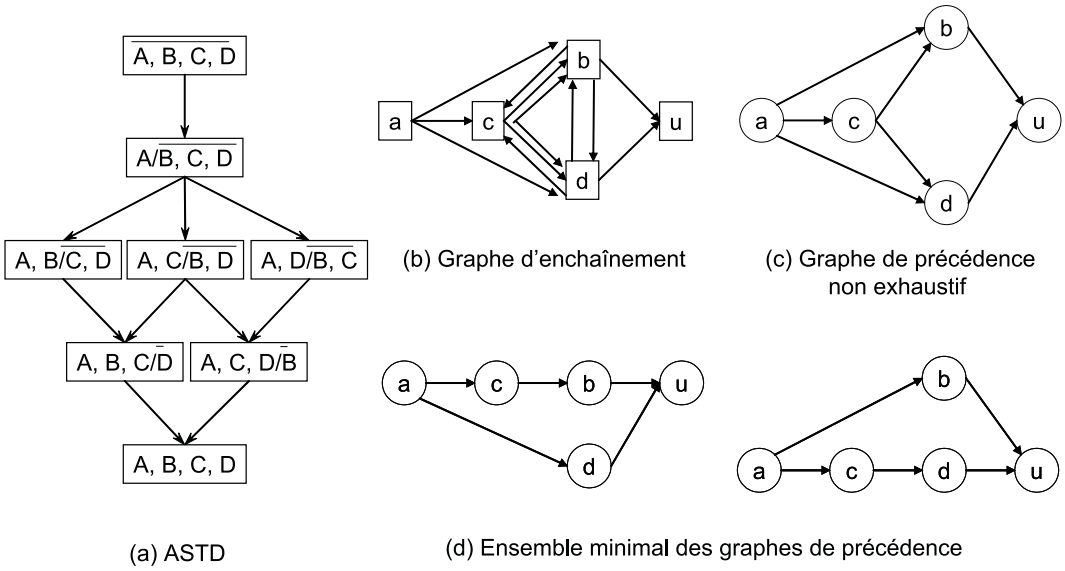


FIG. 3.8 – Cas 4

Sur la figure 3.8, nous pouvons voir que le graphe de précédence, généré avec la méthode de génération de *graphe de précédence*, n'est pas exhaustif, mais il est valide. Cependant il pourrait être encore simplifié, avec la suppression des arcs (a, b) et (a, d) par transitivité.

3.4.5 Évaluation

Nous voyons, dans un certain nombre de cas, que cette méthode est efficace, mais dans un certain nombre d'autres cas, elle engendre des graphes de précédence non exhaustifs, et même quelquefois des graphes de précédence non valides, comme dans le cas de la figure 3.7, où il y a des circuits dans le graphe. D'après la définition 1.17, un graphe de précédence est un graphe simple, orienté, connexe, sans boucle et sans circuit. Alors par définition, tout sommet doit appartenir à une chaîne[13] reliant le sommet initial au sommet final du graphe de précédence.

Pour le cas développé au cours de l'exposition de la méthode et pour les cas 1 et 2, les graphes de précédence engendrés par la méthode sont donc valides et exhaustifs car ceux-ci génèrent les séquences d'enchaînement de l'ensemble de base sans en générer d'autres qui seraient invalides.

[13]Une chaîne est une succession d'arcs, sans tenir compte de leur orientation, entre deux sommets.

En ce qui concerne le cas 3, notre méthode engendre un graphe invalide.

Enfin, pour le cas 4, le graphe de précédence engendré par la méthode est valide, mais n'est pas exhaustif, c'est-à-dire ne permet pas d'obtenir l'ensemble des séquences d'enchaînement initial, mais ne génère pas de séquence d'enchaînement invalide.

De cet exemple nous pouvons donc en déduire les propriétés suivantes :

Propriété 3.6 (Condition de non-validité)
Si Υ l'ensemble des séquences d'enchaînement ne peut pas être représenté par un seul graphe de précédence G alors le graphe de précédence G' généré par notre méthode est non valide ou non exhaustif.
Nous écrivons :

$$\forall \Upsilon, \nexists! G / SEA(G) = \Upsilon \Rightarrow \mathcal{M}(\Upsilon) = G' / SEA(G') \neq \Upsilon$$

Démonstration 3.4
Supposons que notre méthode appliquée sur l'ensemble des séquences $\mathcal{M}(\Upsilon)$ donne un graphe de précédence G' unique représentant Υ ($\mathcal{M}(\Upsilon) = G'/SEA(G') = \Upsilon$) alors il existe un unique graphe de précédence G pour représenter Υ ($\forall \Upsilon, \exists! G / SEA(G) = \Upsilon$). Il y a contradiction donc nous avons :
$\forall \Upsilon, \nexists! G / SEA(G) = \Upsilon, \Rightarrow \mathcal{M}(\Upsilon) = G' / SEA(G') \neq \Upsilon$

Pour conclure l'étude de l'exemple du cas 4, il est donc possible de dire que cette méthode n'est pas efficace dans tous les cas et nous pouvons écrire la propriété de validité suivante :

Propriété 3.7 (Condition de validité)
Si Υ l'ensemble des séquences d'enchaînement peut être représenté par un seul graphe de précédence G alors le graphe de précédence G' généré par notre méthode est valide et exhaustif, il représente l'ensemble Υ des séquences d'enchaînement et est égal au graphe de précédence G.
Nous écrivons :

$$\forall \Upsilon, \exists! G / SEA(G) = \Upsilon \Rightarrow \exists G', \mathcal{M}(\Upsilon) = G' / SEA(G') = \Upsilon, G = G'$$

Démonstration 3.5
Avec les propriétés 3.1, 3.2, 3.5, nous savons que notre méthode donne un graphe unique, si celui-ci est valide, nous avons un graphe de précédence unique.

Supposons qu'il n'existe pas un unique graphe de précédence pour représenter l'ensemble de séquences d'enchaînement initial, comme notre méthode donne toujours un graphe unique G, alors ce graphe ne peut pas représenter l'ensemble des séquences d'enchaînement.
Supposons maintenant que l'ensemble Υ des séquences d'enchaînement peut être représenté par un seul et unique graphe de précédence G. Nous pouvons écrire $\exists! G/SEA(G) = \Upsilon$. Comme notre méthode engendre un graphe de précédence G' unique qui est équivalent au graphe d'enchaînement G_D, étant lui-même équivalent à l'ASTD A, et que l'ASTD généré est équivalent à l'ensemble Υ des séquences d'enchaînement, comme $G = \Upsilon$ et $G' = \Upsilon$ alors $G = G'$

3.5 Améliorations possibles de la méthode

Suite à l'exposé des propriétés 3.6 et 3.7 il est possible de proposer deux améliorations : l'une est pour la génération des graphes de précédence, l'autre pour la génération des hypergraphes de précédence.

3.5.1 Méthode Π-améliorée

Comme nous l'avons vu dans l'exemple ci-avant, notre méthode engendre des graphes de précédence valides et exhaustifs uniquement lorsque l'ensemble initial des séquences d'enchaînement peut être représenté par un seul graphe de précédence. Une amélioration de notre méthode serait *le découpage de l'ensemble des séquences d'enchaînement en sous-ensembles pouvant être représentés respectivement par un seul et unique graphe de précédence.*

Suite aux travaux de A. Bratcu [11], il existe une méthode de découpage des ensembles de séquences d'enchaînement en sous-ensembles pouvant être représentés par un seul graphe de précédence. Pour cela, il suffit d'engendrer à partir de l'ensemble Υ, les sous-ensembles Υ_i tels que chaque Υ_i respecte la propriété Π3.8 introduite par A. Bratcu.

Propriété 3.8 (Π)
Soit Υ un ensemble de séquences d'enchaînement composées de tâches de E. Soit un ensemble $\delta(\Upsilon)$ de paires de tâches de E. On dit que Υ possède la propriété Π par rapport à l'ensemble $\delta(\Upsilon)$ si pour toute séquence x de Υ et pour toute paire de $\delta(\Upsilon)$ en succession di-

recte, la séquence symétrique x' de x, par rapport à la paire considérée, appartient aussi à Υ :

$$\forall x \in \Upsilon, \forall (\alpha_i, \alpha_j) \in \delta(\Upsilon), x = x_{\alpha_i[} \text{ — } \alpha_i \text{ — } \alpha_j \text{— } x_{]\alpha_j} \in \Upsilon$$
$$\Rightarrow \quad x_{\alpha_i[} \text{ — } \alpha_j \text{ — } \alpha_i \text{— } x_{]\alpha_j} \in \Upsilon$$

Définition 3.7 (Séquence d'enchaînement symétrique)
Soit Υ un ensemble de séquences d'enchaînement composées de symboles d'un ensemble $\delta(\Upsilon)$ et soit une séquence d'enchaînement x de Υ. En écrivant x sous la forme : $x = x_{\alpha_i[}$ — α_i — α_j — $x_{]\alpha_j}$,
où $x_{\alpha_i[}$ et $x_{]\alpha_j}$ sont des sous-séquences d'enchaînement de $\delta(\Upsilon)$, (α_i, α_j) étant une paire de symboles de $\delta(\Upsilon)$, la séquence d'enchaînement x' s'écrit : $x' = x_{\alpha_i[}$ — α_j — α_i — $x_{]\alpha_j}$.
x' s'appelle la séquence d'enchaînement symétrique de x par rapport à la paire (α_i, α_j).

À l'aide des exemples des figures 3.5, 3.6, nous voyons que la propriété 3.8 est vérifiée. Nous allons, par exemple, traiter le cas de la figure 3.8, afin d'illustrer notre méthode que nous appellerons «*méthode-*Π*-améliorée*».

L'ensemble Υ_4 des séquences d'enchaînement est :

$$\begin{array}{c} \Upsilon_4 = \{ \\ a \text{ — } c \text{ — } b \text{ — } d \text{ — } u, \\ a \text{ — } c \text{ — } d \text{ — } b \text{ — } u, \\ a \text{ — } b \text{ — } c \text{ — } d \text{ — } u, \\ a \text{ — } d \text{ — } c \text{ — } b \text{ — } u\} \end{array} \tag{3.2}$$

À l'aide de la propriété Π, l'ensemble Υ_4 des séquences d'enchaînement est découpé en deux sous-ensembles Υ_{4_1} et Υ_{4_2}.

$$\begin{array}{c} \Upsilon_{4_1} = \{ \\ a \text{ — } c \text{ — } b \text{ — } d \text{ — } u, \\ a \text{ — } c \text{ — } d \text{ — } b \text{ — } u, \\ a \text{ — } d \text{ — } c \text{ — } b \text{ — } u\} \end{array} \tag{3.3}$$

$$\Upsilon_{4_2} = \{ a \text{ — } b \text{ — } c \text{ — } d \text{ — } u\} \tag{3.4}$$

Avec l'ensemble Υ_{4_1}, l'ASTD $\mathcal{A}_{4_1} = < E_{4_1}, \tau_{4_1} >$ de la figure 3.9 est obtenu :
$E_{4_1} = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, \bar{C}, \bar{D} >, (< A, \bar{B}, \bar{C}, D >, < A, \bar{B}, C, \bar{D} >, < A, \bar{B}, C, D >, < A, B, C, \bar{D} >, < A, B, C, D >\}$
et $\tau_{4_1} = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a), (< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >, d), (< A, \bar{B}, \bar{C}, D >$

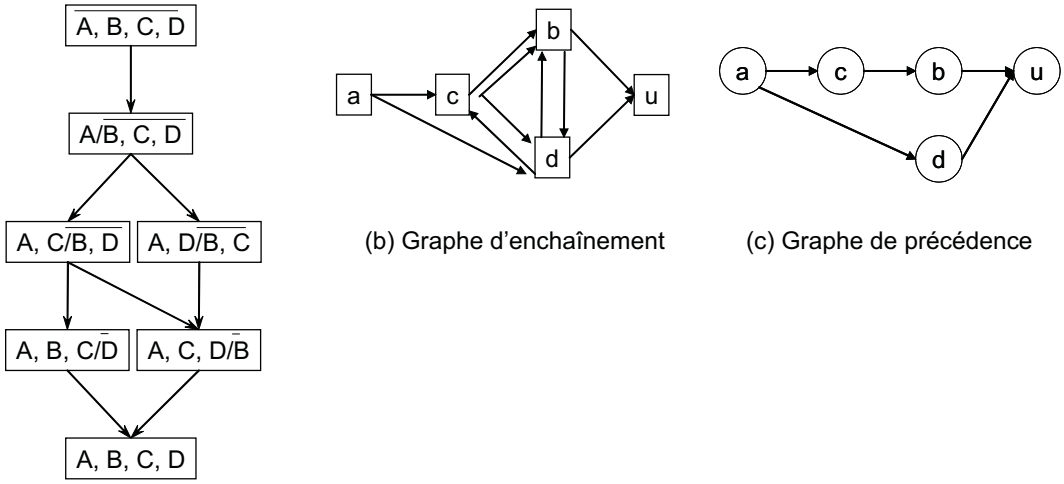


FIG. 3.9 – Cas 4_1 Π-amélioré

$, c), (< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d), (< A, \bar{B}, C, D >, b), (< A, B, C, \bar{D} >, d), (< A, B, C, D >, u)\}$

Les demi-cocycles supérieurs de l'ASTD $\mathcal{A}_{4_1}$ sont :

- $\omega^+_{\mathcal{A}_{4_1}}(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >) = \{(< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, a)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, \bar{B}, \bar{C}, \bar{D} >) = \{(< A, \bar{B}, \bar{C}, \bar{D} >, c), (< A, \bar{B}, \bar{C}, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, \bar{B}, \bar{C}, D >) = \{(< A, \bar{B}, \bar{C}, D >, c)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, \bar{B}, C, \bar{D} >) = \{(< A, \bar{B}, C, \bar{D} >, b), (< A, \bar{B}, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, \bar{B}, C, D >) = \{(< A, \bar{B}, C, D >, b)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, B, C, \bar{D} >) = \{(< A, B, C, \bar{D} >, d)\}$
- $\omega^+_{\mathcal{A}_{4_1}}(< A, B, C, D >) = \{(< A, B, C, D >, u)\}$

Avec le graphe d'enchaînement $G_{D_{4_1}}$ de la figure 3.9, nous avons $G_{D_{4_1}} = (E_{4_1}, \sigma_{4_1})$ où $E_{4_1} = \{a, b, c, d, u\}$ et $\sigma_{4_1} = \{(a, c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (b, d)_{<A,\bar{B},C,\bar{D}>}, (b, u)_{<A,\bar{B},C,D>}, (c, b)_{<A,\bar{B},\bar{C},D>}, (c, b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c, d)_{<A,\bar{B},\bar{C},\bar{D}>}, (d, b)_{<A,\bar{B},C,\bar{D}>}, (d, c)_{<A,\bar{B},\bar{C},\bar{D}>}, (d, u)_{<A,B,C,\bar{D}>}\}$, et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_{4_1}}}(a) = \{(a, c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a, d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4_1}}}(b) = \{(b, d)_{<A,\bar{B},C,\bar{D}>}, (b, u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_{4_1}}}(c) = \{(c, b)_{<A,\bar{B},\bar{C},\bar{D}>}, (c, b)_{<A,\bar{B},\bar{C},D>}, (c, d)_{<A,\bar{B},\bar{C},\bar{D}>}, \}$
- $\omega^+_{G_{D_{4_1}}}(d) = \{(d, b)_{<A,\bar{B},C,\bar{D}>}, (d, c)_{<A,\bar{B},\bar{C},\bar{D}>}, (d, u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_{4_1}}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement non simplifié est :

- $m_{G_{D_{4_1}}}(a,c) = 1$
- $m_{G_{D_{4_1}}}(a,d) = 1$
- $m_{G_{D_{4_1}}}(b,c) = 1$
- $m_{G_{D_{4_1}}}(b,d) = 1$
- $m_{G_{D_{4_1}}}(b,u) = 1$
- $m_{G_{D_{4_1}}}(c,b) = 2$
- $m_{G_{D_{4_1}}}(c,d) = 1$
- $m_{G_{D_{4_1}}}(d,b) = 1$
- $m_{G_{D_{4_1}}}(d,c) = 1$
- $m_{G_{D_{4_1}}}(d,u) = 1$

À l'aide de l'algorithme de simplification, les arcs entre les tâches b et d disparaissent ainsi qu'entre les tâches c et d. Ces tâches deviennent, par ces suppressions respectivement indépendantes.

Nous obtenons alors le graphe d'enchaînement simplifié suivant :

$G_{D_{4s_1}} = (E_{4s_1}, \sigma_{4s_1}, \xi_{4s_1}, L_{4s_1})$ où
$E_{4s_1} = \{a, b, c, d, u\}$
et $\sigma_{4s_1} = \{(a,c), (a,d), (b,u), (c,b), (d,u)\}$,
$\sigma_{4s_1} = \{< \bar{A}, \bar{B}, \bar{C}, \bar{D} >, < A, \bar{B}, C, D >, < A, \bar{B}, \bar{C}, D >, < A, B, C, \bar{D} >\}$,
et $L_{4s_1} = \{((a,c), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((a,d), < \bar{A}, \bar{B}, \bar{C}, \bar{D} >), ((b,u), < A, \bar{B}, C, D >)$,
$((c,b), < A, \bar{B}, \bar{C}, D >), ((d,u), < A, B, C, \bar{D} >)\}$,
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_{4s_1}}}(a) = \{(a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4s_1}}}(b) = \{(b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_{4s_1}}}(c) = \{(c,b)_{<A,\bar{B},\bar{C},\bar{D}>}, \}$
- $\omega^+_{G_{D_{4s_1}}}(d) = \{(d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_{4s_1}}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement simplifié est :

- $m_{G_{D_{4s_1}}}(a,c) = 1$
- $m_{G_{D_{4s_1}}}(a,d) = 1$
- $m_{G_{D_{4s_1}}}(b,u) = 1$
- $m_{G_{D_{4s_1}}}(c,b) = 1$
- $m_{G_{D_{4s_1}}}(d,u) = 1$

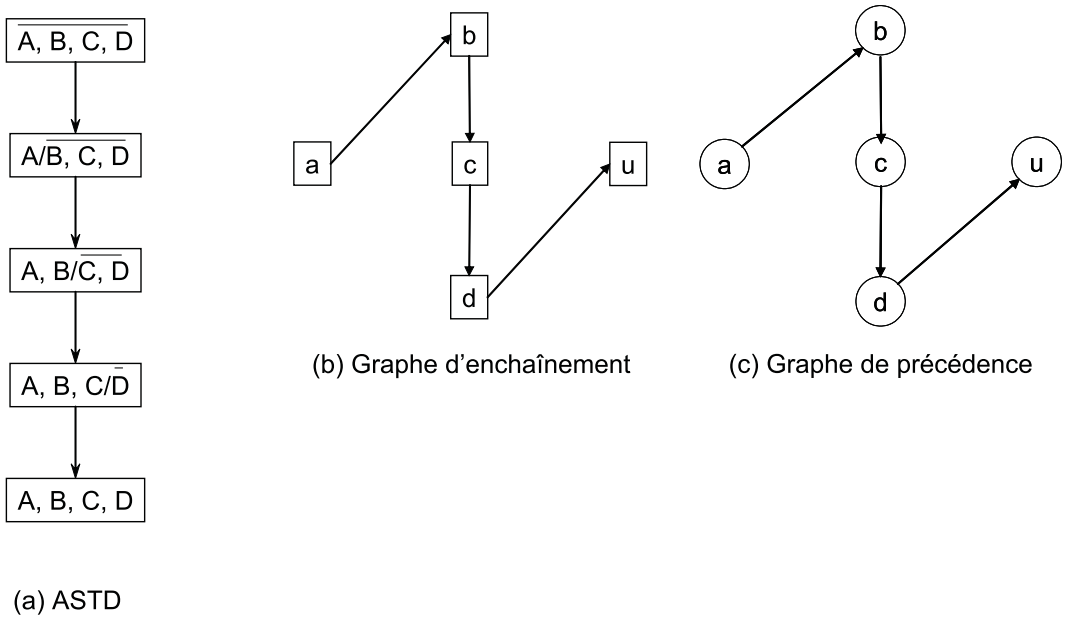


FIG. 3.10 – Cas 4_2 Π-amélioré

Avec la méthode Π-améliorée, le graphe de précédence de la figure 3.9 est obtenu. Ce graphe représente de manière biunivoque l'ensemble Υ_{4_1} des séquences d'enchaînement.

Maintenant, nous allons passer au deuxième sous-ensemble Υ_{4_2} de Υ_4. Avec l'ensemble Υ_{4_2} qui est identique à l'ensemble Υ_1, si nous nous rapportons à la section 3.4.1, nous voyons que le graphe de précédence de la figure 3.10 est valide, exhaustif et correspond de manière biunivoque à l'ensemble Υ_{4_2}.

Pour conclure sur cette amélioration, il est possible de dire que cette méthode est efficace pour la génération des graphes de précédence valides et exhaustifs. Cependant le découpage de l'ensemble des séquences d'assemblage en sous-ensembles représentables par un seul graphe de précédence n'est pas trivial. C'est pour cela que nous allons nous pencher sur une autre méthode d'amélioration.

3.5.2 Détermination de l'hypergraphe de précédence

Après simplification des arcs comme nous l'avons vu dans la section 3.3, un graphe d'enchaînement est obtenu. Deux cas peuvent se produire : le premier est le plus simple, le graphe d'enchaînement simplifié est équivalent au graphe de précédence associé à l'en-

semble Υ de départ ; le deuxième est plus complexe, le graphe d'enchaînement simplifié est dit *complexe*, il est équivalent à un hypergraphe de précédence.

Dans un graphe d'enchaînement complexe, il y a présence d'arcs opposés ou d'arc(s) complexe(s)[14] entre deux sommets.

La détermination d'un hypergraphe de précédence est due au fait que l'ensemble Υ des séquences d'enchaînement ne peut pas être représenté par un seul graphe de précédence. Ce qui traduit la présence de conditions de précédence du type la tâche a précède la tâche b ou la tâche c précède la tâche b.

Pour cela en nous inspirant de T. De Fazio et D.E. Whitney [14], nous supposerons que toutes les contraintes de précédence peuvent s'écrire sous la forme :

$$f(E - \alpha_i) \rightarrow \alpha_i \text{ et } \alpha_i \rightarrow g(E - \alpha_i) \tag{3.5}$$

Où : E est l'ensemble des tâches, f et g deux fonctions logiques.

Les fonctions logiques peuvent être écrites sous leur forme normale conjonctive de clauses, où chaque clause est une tâche d'assemblage ou une disjonction de contrainte de précédence :

$$(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge A_1 \wedge A_2 \wedge \ldots) \rightarrow \alpha_i \text{ et } \alpha_i \rightarrow (\beta_1 \wedge \beta_2 \wedge \ldots \wedge B_1 \wedge B_2 \wedge \ldots) \tag{3.6}$$

Où :$\alpha_1, \alpha_2, \ldots, \beta_1, \beta_2, \ldots$ sont des tâches d'assemblage et $A_1, A_2, \ldots, B_1, B_2, \ldots$ sont des disjonctions de tâches d'assemblage.

Détermination des précédences conditionnelles Pour l'ensemble de ces situations, nous devons définir ce que sont les *précédences conditionnelles.*

Définition 3.8 (Précédence conditionnelle)
Nous appelons précédence conditionnelle une précédence qui n'est valide que dans certaines conditions.

Préalablement nous introduirons la propriété suivante :

[14] Arc(s) orienté(s) entre les tâches α_i et α_j avec indifférence entre les tâches α_i et α_j par rapport à une autre paire d'arcs

Propriété 3.9 (Dualité des précédences conditionnelles)
Si dans un graphe d'enchaînement, nous avons plus d'arcs, entre deux sommets i et j, dans un sens que dans l'autre, alors il existe une autre paire de sommets dans les mêmes conditions.

$$\forall G_D, \exists i,j \in G_D | m_{G_D}(i,j) \neq m_{G_D}(j,i) \Rightarrow \exists k,l \in G_D | m_{G_D}(k,l) \neq m_{G_D}(l,k)$$

Démonstration 3.6
Comme chaque séquence d'enchaînement est un ordre total sur l'ensemble des tâches, elle est représentée par un chemin hamiltonien dans le graphe d'enchaînement, alors chaque sommet de ce graphe appartient à chaque séquence. S'il existe plusieurs arcs entre les sommets i et j, avec plus d'arcs[15]*, par exemple, de i vers j que de j vers i, et de plus que l'état initial de chacun de ces arcs est différent, alors il existe, dans ce graphe, une paire de sommets (k,l) ayant plus d'arcs, par exemple, du sommet k au sommet l que du sommet l au sommet k*[16].

Les précédences conditionnelles sont exprimées dans le graphe d'enchaînement par deux types de relations entre les nœuds : soit il reste au moins une précédence d'une tâche sur une autre avec des paires d'arcs entre ces deux tâches supprimées, soit il existe plusieurs arcs opposés d'états initiaux différents entre ces deux tâches.

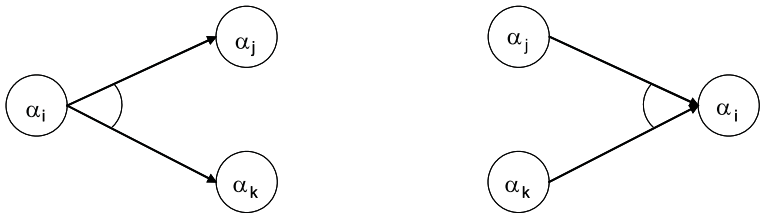
L'écriture des précédences conditionnelles est simple, il suffit de relever toutes les relations entre les tâches vérifiant les deux cas cités ci-avant.

Avec l'exemple traité à la section 3.4.3, nous avons les deux séquences d'enchaînement $a — b — c — d — u$ et $a — d — c — b — u$. À l'aide du graphe d'enchaînement simplifié de la figure 3.7, nous voyons qu'il y a des paires d'arcs opposés non simplifiées dans le graphe entre les sommets b et c et entre les sommets c et d. Dans ce cas, nous avons donc une condition sur la tâche c, qui doit être effectuée entre la tâche b et la tâche d. Ces précédences conditionnelles sont notées :

$$\begin{aligned} &b\,Pc\,c \\ &c\,Pc\,d \\ &d\,Pc\,c \\ &c\,Pc\,b \end{aligned} \tag{3.7}$$

Après l'écriture de l'ensemble de ces précédences conditionnelles, nous devons définir à quels arcs appartient chaque précédence conditionnelle.

[15] C'est à dire qu'il y a plus de séquences d'enchaînement où la tâche i précède la tâche j que l'inverse.
[16] Il est possible que k ou l soit égal à i ou à j, mais dans aucun cas les deux en même temps.



(a) Hypergraphe aval (b) Hypergraphe amont

FIG. 3.11 – Différents hypergraphes

Pour cela, il est nécessaire de définir la propriété suivante :

PROPRIÉTÉ 3.10 (REGROUPEMENT DES PRÉCÉDENCES CONDITIONNELLES)
Pour un ensemble Υ de séquences d'enchaînement, si les tâches a_i et a_j et a_k (avec $i \neq j$ et $i \neq k$) sont en relation de précédence conditionnelle, soit la tâche α_i précède la tâche α_j ou la tâche α_k, soit la tâche α_j ou la tâche α_k précèdent la tâche α_i, il est possible de générer les sous-ensembles $\Upsilon_{aval}(a_i)$ et $\Upsilon_{amont}(\alpha_i)$ tels que quelque soit la tâche α_n de $\Upsilon_{aval}(\alpha_i)$, nous ayons toujours α_i précède conditionnellement la tâche α_n et quelque soit la tâche α_m de $\Upsilon_{amont}(\alpha_i)$, nous ayons toujours α_m précède conditionnellement la tâche α_i.

$$\forall \alpha_i, \exists \Upsilon_{aval}(\alpha_i) | \forall \alpha_j \in \Upsilon_{aval}(\alpha_i), \alpha_i \ Pc \ \alpha_j,$$
$$\forall \alpha_i, \exists \Upsilon_{amont}(\alpha_i) | \forall \alpha_k \in \Upsilon_{amont}(\alpha_i), \alpha_k \ Pc \ \alpha_i$$

Avec la propriété 3.10, il est alors possible de séparer l'ensemble des précédences conditionnelles en sous-ensembles : l'un portant sur les hyperarcs aval et l'autre sur les hyperarcs amont (voir figure 3.11).

Il est donc possible avec l'exemple ci-avant d'écrire :

$$\begin{aligned} \Upsilon_{aval}(c) &= \{b \text{ Pc } c, d \text{ Pc } c\} \\ \Upsilon_{amont}(c) &= \{c \text{ Pc } d, c \text{ Pc } b\} \end{aligned} \tag{3.8}$$

Détermination des hyperarcs de précédence Après la détermination des deux groupes de précédences conditionnelles $\Upsilon_{aval}(\alpha_i)$ et $\Upsilon_{amont}(\alpha_i)$, nous devons déterminer les hyperarcs de l'hypergraphe de précédence. Avant de passer à cette détermination des hyperarcs de précédence, nous posons l'hypothèse suivante :

HYPOTHÈSE 3.1

Nous ne considérons que des ensembles Υ_{aval} et Υ_{amont} de deux éléments maximum.

Soit :

$$\forall \Upsilon, \forall \alpha_i, Card(\Upsilon_{aval}(\alpha_i) \leq 2), Card(\Upsilon_{amont}(\alpha_i) \leq 2)$$

Pour cela, nous devons générer les sous-ensembles des séquences terminales avales, qui sont notés $\Upsilon_{s_{aval}}(\alpha_i)$, c'est à dire la portion de séquence d'enchaînement comprenant les tâches de l'intersection $\beta_{\cap}(\alpha_i)$ de l'ensemble β_{α_i} de toutes les tâches intervenant après la tâche a_i dans l'ensemble des séquences d'enchaînement et de l'ensemble de toutes les tâches appartenant à $\Upsilon_{aval}(\alpha_i)$.

$\Upsilon_{s_{aval}}(\alpha_i)$ représente alors l'ensemble des successions de tâches suivant la tâche α_i tout en étant en relation de précédence conditionnelle avec la tâche α_i.

Dans le cas de l'exemple de la figure 3.7, nous avons les ensembles $\Upsilon_{aval}(c) = \{c \text{ Pc } b \text{ ,} c \text{ Pc } d\}$ et $\beta_c = \{b, d, u\}$, qui nous donnent l'ensemble $\beta_{\cap}(c) = b, d$. Nous avons alors $\Upsilon_{s_{aval}}(c) = \{c — d, c — b\}$.

À l'aide de la propriété 3.10, nous générons les précédences conditionnelles (3.9) et (3.10) en les séparant en deux groupes *aval* et *amont*. Les précédences conditionnelles (3.9) représentent un ensemble de branches d'hypergraphes de type *aval* :

$$\begin{array}{l} b — c \\ d — c \end{array} \tag{3.9}$$

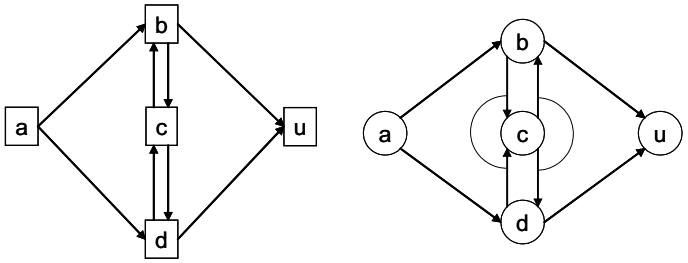
Il existe bien entendu les mêmes ensembles pour les hyperarcs amont.

Les précédences conditionnelles (3.10) représentent un ensemble de branches d'hypergraphes de type *amont* :

$$\begin{array}{l} c — d \\ c — b \end{array} \tag{3.10}$$

Une fois les sous-ensembles des séquences terminales établis, il est possible de générer pour chaque sommet l'ensemble des disjonctions des précédences possibles. Nous faisons une disjonction de tous les éléments de l'ensemble $\Upsilon_{aval}(\alpha_i)$. Un ou plusieurs hyperarcs sont donnés après transformation de la forme normale disjonctive en une forme normale conjonctive.

Dans un cas simple de deux précédences conditionnelles, les possibilités sont réduites à un seul cas alors l'hyperarc est facilement obtenu. Il peut se représenter par la figure 3.12(b).



(a) Graphe d'enchaînement (b) Hypergraphe de précédence

FIG. 3.12 – Cas 3 représenté par hypergraphe

3.5.3 Exemple

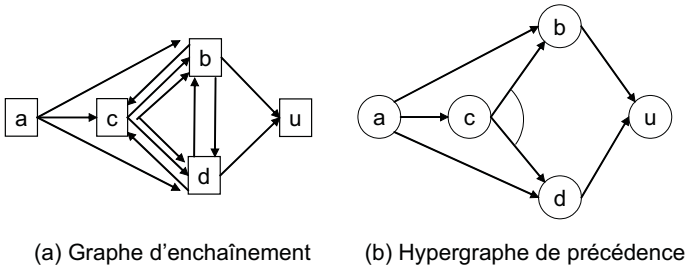
Pour cet exemple, nous reprenons les deux cas 3 et 4 où nous n'avions pas obtenu des graphes de précédence valides et exhaustifs. Le cas 3 a été développé au cours de l'exposition de la méthode, nous allons donc passer au cas 4.

Dans le cas 4, après simplification nous avons le graphe d'enchaînement $G_{D_{4s}}$:
$G_{D_{4s}} = (E_{4s}, \sigma_{4s}, \xi_{4s}, L_{4s})$ où
$E_{4s} = \{a, b, c, d, u\}$,
$\sigma_{4s} = \{(a,b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (b,u)_{<A,\bar{B},C,D>}, (c,b)_{<A,\bar{B},\bar{C},D>}$,
$(c,d)_{<A,B,\bar{C},\bar{D}>}, (d,u)_{<A,B,C,\bar{D}>}\}$,
$\xi_{4s} = \{<\bar{A},\bar{B},\bar{C},\bar{D}>, <A,\bar{B},\bar{C},\bar{D}>, <A,\bar{B},\bar{C},D>, <A,B,\bar{C},\bar{D}>, <A,\bar{B},C,\bar{D}>$
$, <A,\bar{B},C,D>, <A,B,C,\bar{D}>, <A,B,C,D>\}$
et $L_{4s} = \{((a,b), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((a,c), <\bar{A},\bar{B},\bar{C},\bar{D}>), ((a,d), <\bar{A},\bar{B},\bar{C},\bar{D}>)$,
$((b,u), <A,\bar{B},C,D>), ((c,b), <A,\bar{B},\bar{C},D>), ((c,d), <A,B,\bar{C},\bar{D}>)$,
$((d,u), <A,B,C,\bar{D}>)\}$,
et les demi-cocycles supérieurs sont :

- $\omega^+_{G_{D_{4s}}}(a) = \{(a,b)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,c)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}, (a,d)_{<\bar{A},\bar{B},\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(b) = \{(b,u)_{<A,\bar{B},C,D>}\}$
- $\omega^+_{G_{D_{4s}}}(c) = \{(c,b)_{<A,\bar{B},\bar{C},D>}, (c,d)_{<A,B,\bar{C},\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(d) = \{(d,u)_{<A,B,C,\bar{D}>}\}$
- $\omega^+_{G_{D_{4s}}}(u) = \emptyset$

Le nombre d'arcs entre chaque paire de sommets du graphe d'enchaînement simplifié est :

- $m_{G_{D_{4s}}}(a,b) = 1$
- $m_{G_{D_{4s}}}(a,c) = 1$
- $m_{G_{D_{4s}}}(a,d) = 1$



(a) Graphe d'enchaînement (b) Hypergraphe de précédence

FIG. 3.13 – Cas 4 représenté par hypergraphe

- $m_{G_{D_{4s}}}(b, u) = 1$
- $m_{G_{D_{4s}}}(c, b) = 1$
- $m_{G_{D_{4s}}}(c, d) = 1$
- $m_{G_{D_{4s}}}(d, u) = 1$

À l'aide de notre méthode, nous voyons que les tâches b et c sont indifférentes et nous pouvons voir aussi que c précède conditionnellement b et précède conditionnellement aussi c. De là nous générons les deux groupes de précédences conditionnelles *amont* et *aval*. Dans ce cas, il n'y a des éléments que dans le sous-ensemble aval, qui est :

$$\begin{aligned} c\,Pcb \\ c\,Pcd \end{aligned} \tag{3.11}$$

Dans ce cas très simple, si nous n'avons que deux arcs, il n'y a pas de problème de répartition des précédences conditionnelles dans les hyperarcs.

L'hypergraphe ainsi obtenu peut se représenter par la figure 3.13(b).

Après la présentation de ces diverses améliorations, nous allons estimer la complexité de notre méthode.

3.6 Complexité

Afin de calculer la complexité de l'algorithme de transformation de l'ensemble des séquences d'enchaînement en un graphe ou hypergraphe de précédence, nous devons définir quelles étapes rentrent dans ce calcul et lesquelles ne rentrent pas dedans.

Pour notre calcul de complexité, nous tiendrons compte du nombre de balayage des ensembles de séquences, du nombre de balayage de chaque séquence ou chaque graphe.

Nous ne tiendrons pas compte des tâches unitaires comme des affectations, des comparaisons de deux précédences, génération d'un état à partir d'un autre et d'une tâche

ou encore insertion d'une tâche dans l'ensemble des tâches d'un graphe. Par exemple, toutes ces tâches comme par exemple génération d'un état, insertion de cet état et de la tâche associée sont regroupées en une seule tâche au niveau de ce calcul de complexité.

Nous avons un produit $\mathcal{P}$ composé de p constituants. L'ensemble Υ des séquences d'enchaînement comporte n séquences valides.

Pour la génération d'un ASTD, qui comporte deux phases récurrentes, nous avons un seul balayage de l'ensemble Υ des n séquences d'enchaînement comportant toutes $p+1$ termes. La complexité de l'algorithme de génération d'un ASTD est $O(n \times p)$.

Comme nous l'avons vu dans le chapitre 2, un ASTD ayant un seul composant de base est composé de $1+2^p$ nœuds au maximum avec seulement $p-1$ nœuds par niveau. Comme ce graphe est équivalent à l'ensemble Υ son parcours se fait en n cycles. Le graphe d'enchaînement est alors généré en $n \times p$ tâches. La complexité de l'algorithme de génération du graphe d'enchaînement est $O(n \times p)$.

La phase génération du graphe de précédence est composée de deux grandes étapes, la *simplification du graphe d'enchaînement* $\mathcal{M}_{G_s}$ et la *détermination du graphe de précédence* $\mathcal{M}_{G_d}$. Comme la simplification d'un graphe d'enchaînement se fait par paire de sommets et que nous possédons les demi-cocycles inférieurs et supérieurs, nous pouvons réduire cet ordre de complexité à un cycle sur chaque paire possible de sommets ; c'est à dire $(p+1) * p/2$ possibilités. La complexité de l'algorithme de simplification est $O(\frac{(p+1)\times p}{2})$ il est possible d'approximer par $O(\frac{p^2}{2})$

Si nous générons un graphe de précédence, la détermination du graphe de précédence est une recopie des ensembles des nœuds et des enchaînements. L'ensemble des nœuds comporte toujours $p+1$ éléments, tandis que l'ensemble des enchaînements peut contenir entre p et $2 \times (p-1)$ arcs. L'algorithme de détermination de graphe de précédence a une complexité de $O(3p)$.

Avec les complexités respectives des algorithmes de cette méthode, nous avons une complexité de génération globale des graphes de précédence de $O(p^2+2np)$ dans les cas favorables.

3.7 Conclusion du chapitre

Ce chapitre a été dédié à l'étude d'une nouvelle méthode de génération systématique des graphes de précédence, à partir d'un ensemble de séquences d'enchaînement pré-

établies avec LEGA, où chacune d'elles est linéaire. Pour ce faire, nous avons utilisé les ASTD afin de connaître l'ensemble des états possibles de l'assemblage du produit $\mathcal{P}$ considéré. Puis, avec l'ASTD, nous engendrons son graphe dual, que nous appelons graphe d'enchaînement. De cet enchaînement naît une notion de précédence, qui s'obtient par simplification successive des paires d'arcs inverses de même état initial, entre deux tâches, c'est-à-dire s'il existe un arc (α_i, α_j) d'état initial $\mathcal{E}$ et un arc (α_j, α_i) de même état initial $\mathcal{E}$ alors nous les supprimons. Après la simplification de toutes les paires d'arcs possibles, nous obtenons un graphe de précédence valide et exhaustif si et seulement si l'ensemble des séquences d'enchaînement peut être représenté par un seul graphe de précédence. Dans ce cas, cette méthode conduit simplement et directement à un graphe de précédence. Dans le cas contraire, un hypergraphe de précédence peut être engendré. La notion d'hypergraphe de précédence est intéressante, de par sa capacité à représenter un ensemble de graphes de précédence.

Pour traiter ces autres cas, nous proposons deux méthodes. La première d'entre elles repose sur la propriété Π de A. Bratcu [11], pour découper l'ensemble des séquences d'enchaînement en sous-ensembles de séquences permettant d'engendrer un graphe de précédence unique, cependant la complexité de l'algorithme de découpage ne nous permet pas de dire que ce soit une méthode simple et efficace. La deuxième méthode d'amélioration a été proposée pour générer les hypergraphes de précédence, dans le carde de l'hypothèse de T. De Fazio et D. E. Whitney [14], c'est à dire que nous pouvons écrire les contraintes de précédence sous la forme $f(\alpha_j) \rightarrow \alpha_i$ et $\alpha_i \rightarrow g(\alpha_j)$, avec $j \neq i$ et f et g deux fonctions logiques. De plus nous ne traitons que des cas simplifiés où les deux fonctions logiques ne portent que sur deux variables. Elle est composée de deux grandes étapes, la première est la détermination des précédences conditionnelles, la deuxième est la détermination des hyperarcs de précédence. Par un découpage de l'ensemble des précédences conditionnelles, nous obtenons des hyperarcs de précédence, puis de là en découle des hypergraphes de précédence.

Comme nous avons pu le voir, nous pouvons générer des graphes de précédence et des hypergraphes de précédence à l'aide de notre méthode. Cependant, cette méthode ne nous satisfait pas complètement au niveau de la génération des hypergraphes de précédence, nous allons donc passer le chapitre suivant à mettre en place une autre méthode de génération systématique des graphes de précédence. La méthode qui sera proposée, reprendra les hypothèses établies dans le chapitre précédent, et sera principalement basée sur la logique booléenne et permettra de générer des graphes de précédence et des hypergraphes de précédence.

Chapitre 4

Détermination des graphes de précédence par la logique booléenne

Comme il a été précisé, dans les chapitres précédents, les graphes de précédence sont très utilisés en conception de systèmes d'assemblage. De plus actuellement, l'absence de méthode suffisamment simple d'utilisation et efficace pour la génération des graphes de précédence, nous a poussé à développer dans ce chapitre, une nouvelle méthode de génération systématique des graphes de précédence.

4.1 Introduction

K.S. Naphade exprime les relations de précédence à l'aide des opérateurs logiques de la manière suivante : $((a \rightarrow c)$ and $((b \rightarrow c)$ or $d \rightarrow c))$ or $((e \rightarrow c)$ and $(f \rightarrow c)))$. Une présentation plus détaillée est proposée en annexe B.

Cette expression introduit les notions de *disjonction de contraintes de précédence*[1] et de *conjonction de contraintes de précédence*[2]. Selon P. De Lit [15], K.S. Naphade a commis une «*erreur*» dans la définition du *complémentaire d'une précédence* (voir Annexe B). Cette annexe a été introduite pour résoudre un problème persistant dans la méthode de K.S. Naphade, et ce malgré les travaux de P. De Lit.

Suite à ces travaux, nous avons mis en place une méthode de génération des graphes de précédence à l'aide d'un outil mathématique : *la logique booléenne*. Cet outil est une

[1] Opérateur correspondant à un «ou» logique entre des contraintes de précédence.
[2] Opérateur correspondant à un «et» logique entre des contraintes de précédence.

transformation qui permet de convertir des séquences d'enchaînement en un ou plusieurs graphes de précédence, en passant par une expression logique des contraintes de précédence, qui est un ensemble de précédences mises en relation par différents opérateurs[3].

Remarque 4.1
Un graphe de précédence est équivalent à un ensemble conjonctif de contraintes de précédence.

Afin de mieux appréhender cette transformation, quelques notions de base sur la *théorie des ensembles* sont rappelées, puis la méthode de *génération des graphes de précédence* sera développée, les différentes règles de résolution de notre problème seront alors précisées. Pour clôturer ce chapitre, nous étudierons de la complexité de la méthode proposée.

4.2 Notions de base et notations.

Quelques définitions sont développées ici, tout d'abord sur la théorie des ensembles en général, puis sur la théorie des ensembles des séquences d'enchaînement. Quelques notions sur les séquences d'enchaînement seront précisées et un exemple illustrera ces différents points.

4.2.1 Théorie des ensembles

Dans cette section, nous définirons ce qu'est un ensemble, ainsi que la notion d'appartenance. Il serait impossible de parler d'appartenance d'un élément à un ensemble sans l'existence de la notion de comparaison de deux éléments.

Les définitions suivantes sont tirées des travaux de M. Queysannes [47] :

Définition 4.1 (Comparaison)
Deux objets sont considérés comme bien définis si nous possédons un critère permettant d'affirmer que deux de ces objets représentés par a et par b sont identiques *ou* distincts, *notés :*

$$a = b \qquad ou \qquad a \neq b$$

[3] Ce sont les opérateurs logiques booléens courants : «ou» noté «+», «et» noté «·», la négation d'une contrainte de précédence a été introduite dans les hypothèses du chapitre 2, toutefois les opérateurs «ou» et «et» sont notés officiellement respectivement «∨» et «∧» mais cette notation est lourde.

Avec la définition 4.1, il est possible d'écrire la définition suivante de l'ensemble.

Définition 4.2 (Ensemble)
Un ensemble E *est bien défini lorsqu'on possède un critère permettant d'affirmer pour tout objet* a*, s'il appartient à l'ensemble* E *ou n'appartient pas à l'ensemble* E *; on écrit :*

$$a \in E$$
$$a \notin E$$

Avec l'exemple trivial de l'ensemble[4] $\{a, b, c\}$ et de l'élément a, les définitions 4.1 et 4.2 permettent de dire que a appartient à l'ensemble $\{a, b, c\}$, cette appartenance est notée $a \in \{a, b, c\}$

Définition 4.3 (Ensemble fini)
Un ensemble E *est* fini*, s'il est vide ou s'il existe une bijection* e *sur un intervalle fermé* $[1, a]$ *de* $\mathbb{N}$ $(a \neq 0)$

Nous pouvons dire qu'un ensemble est fini si nous pouvons énumérer tous ses éléments.

Tout ensemble peut être décrit par sa fonction caractéristique.

Définition 4.4 (Fonction caractéristique)
Une fonction booléenne $\mathcal{F}$ *est la fonction caractéristique d'un ensemble* E *si quelque soit* e *un élément, la fonction* $\mathcal{F}$ *appliquée à l'élément* e *est vraie si et seulement si l'élément* e *appartient à l'ensemble* E*.*

$$\forall E, \exists \mathcal{F} : E \mapsto \{\ Faux,\ Vrai\} | \forall e, \mathcal{F}(e) = \ Vrai \ \Leftrightarrow e \in E$$

Avant d'appliquer ces définitions aux ensembles des séquences d'enchaînement, nous devons introduire une propriété de ces séquences.

4.2.2 Séquence d'enchaînement

Des hypothèses sur les séquences d'enchaînement ont été introduites au chapitre 2 pour simplifier le problème de génération des graphes de précédence. Une séquence d'enchaînement est une succession totalement ordonnée de tâches d'assemblage, ce qui exclut donc tout parallélisme.

[4] Par définition, un ensemble ne possède pas deux fois le même élément.

Une séquence d'enchaînement x_2 est un ensemble de permutations des tâches d'une autre séquence d'enchaînement x_1 du même produit, comme le montre l'expression suivante :

$$\begin{aligned} x_1 &= \alpha_1 — \alpha_2 — \alpha_3 \ldots \alpha_i — \alpha_j \ldots \alpha_{n-2} — \alpha_{n-1} — \alpha_n \\ x_2 &= \alpha_1 — \alpha_2 — \alpha_3 \ldots \alpha_j — \alpha_i \ldots \alpha_{n-2} — \alpha_{n-1} — \alpha_n \end{aligned} \tag{4.1}$$

À partir de cette notation, il est possible d'écrire la propriété suivante :

Propriété 4.1 (Séquence de précédences)
Toute séquence d'enchaînement peut s'écrire sous la forme de conjonctions de précédences, elle sera alors appelée séquence de précédences.

$$x \equiv \alpha_1 \rightarrow \alpha_2 \cdot \alpha_2 \rightarrow \alpha_3 \cdot \ldots \cdot \alpha_i \rightarrow \alpha_j \cdot \ldots \cdot \alpha_{n-1} \rightarrow \alpha_n \equiv \prod_{i=1}^{n-1} (\alpha_i \rightarrow \alpha_{i+1})$$

Où les symboles «·» et «Π» représentent l'opérateur logique «et».

Démonstration 4.1
Comme il a déjà été précisé, l'enchaînement de deux tâches est représenté, dans une séquence d'enchaînement, par le symbole «—». Lorsque la séquence d'enchaînement s est composée de deux tâches α_i et α_j, nous avons $x = \alpha_i — \alpha_j$. Trivialement, il est possible de transcrire cette séquence en la contrainte de précédence suivante : α_i précède α_j, notée $\alpha_i \rightarrow \alpha_j$.
Il est évident que si $x = \alpha_i — \alpha_j — \alpha_k$, nous avons α_i précède α_j qui précède α_k alors il est possible d'en déduire que α_i précède α_j et que α_j précède α_k noté alors $\alpha_i \rightarrow \alpha_j \cdot \alpha_j \rightarrow \alpha_k$.
Cette écriture peut être généralisée. Une séquence d'enchaînement $x = \alpha_1 — \alpha_2 — \alpha_3 — \ldots — \alpha_i — \alpha_j — \ldots — \alpha_{n-1} — \alpha_n$ peut alors se décomposer en une séquence de précédences (4.2).

$$x = \alpha_1 \rightarrow \alpha_2 \cdot \alpha_2 \rightarrow \alpha_3 \cdot \ldots \cdot \alpha_i \rightarrow \alpha_j \cdot \ldots \cdot \alpha_{n-2} \rightarrow \alpha_{n-1} \cdot \alpha_{n-1} \rightarrow \alpha_n \tag{4.2}$$

Par habitude et pour simplifier la notation, l'expression (4.2) est notée :

$$x = \prod_{i=1}^{n-1} (\alpha_i \rightarrow \alpha_{i+1}) \tag{4.3}$$

Comme une séquence d'enchaînement est un ensemble de permutations d'une autre séquence d'enchaînement, nous pouvons facilement transcrire ces permutations au sein

d'une séquence de précédence par la négation d'une précédence et écrire la propriété suivante :

Propriété 4.2 (Enchaînement)
Dans une séquence d'enchaînement donnée, la complémentarité d'un enchaînement de deux tâches peut s'écrire :

$$\neg(\alpha_i \rightarrow \alpha_j) \equiv \alpha_j \rightarrow \alpha_i.$$

Cette propriété est triviale puisqu'une séquence définit un ordre total donc, soit a précède b (noté $a \rightarrow b$) soit b précède a (noté $b \rightarrow a$).

4.2.3 Ensemble de séquences d'enchaînement

Sachant que notre travail porte sur la génération des graphes de précédence à partir des séquences d'enchaînement valides préalablement établies par LEGA, il est possible d'écrire les définitions suivantes inspirées des définitions 4.1, 4.2, 4.3 et 4.4.

Définition 4.5 (Comparaison des séquences d'enchaînement)
Les séquences d'enchaînement sont considérées comme bien définies si nous possédons un critère permettant d'affirmer que deux de ces séquences d'enchaînement, représentées par x_1 et par x_2, sont identiques *ou* distinctes *; nous écrirons :*

égalité : $x_1 = x_2$ *si et seulement si* tous *les enchaînements de la séquence x_1 sont* identiques *à* tous *les enchaînements de la séquence x_2 ;*

différence : $x_1 \neq x_2$ *si et seulement s'il y a* au moins un *enchaînement de la séquence x_1* en contradiction *avec l'un des enchaînements de la séquence x_2 ; c'est-à-dire par exemple si $x_1 = \alpha_1 — \alpha_2$ et que $x_2 = \alpha_2 — \alpha_1$.*

Définition 4.6 (Ensemble de séquences d'enchaînement)
Un ensemble Υ *de séquences d'enchaînement est bien défini lorsqu'on possède un critère permettant d'affirmer pour toute séquence d'enchaînement x, si elle appartient à l'ensemble Υ ou n'appartient pas à l'ensemble Υ ; on écrit :*

$$x \in \Upsilon$$
$$x \notin \Upsilon$$

Avec la définition4.6, il est possible de dire que l'ensemble des séquences d'enchaînement est fini.

Définition 4.7 (Fonction caractéristique d'un ensemble de séquences)
Une fonction booléenne $\mathcal{F}$ est la fonction caractéristique d'un ensemble Υ de séquences d'enchaînement valides x_i si, quelque soit x une séquence d'enchaînement, la fonction $\mathcal{F}$ appliquée à la séquence d'enchaînement x est vraie si et seulement si la séquence d'enchaînement x appartient à l'ensemble Υ.

$$\forall \Upsilon, \exists \mathcal{F} : \Upsilon \mapsto \{Faux,\ Vrai\} | \forall x, \mathcal{F}_\Upsilon(x) = Vrai \Leftrightarrow x \in \Upsilon \tag{4.4}$$

Comme nous savons que tout ensemble peut être représenté par une fonction caractéristique, il est possible d'écrire la propriété suivante :

Propriété 4.3 (Fonction caractéristique)
La fonction caractéristique $\mathcal{F}(x)$ d'un ensemble Υ de séquences d'enchaînement valides x_i peut s'écrire :

$$\mathcal{F}_\Upsilon(x) = (x = x_1) + (x = x_2) + \ldots + (x = x_n) = \sum_{i=1}^{n} (x = x_i)$$

Où les symboles «+» et «Σ» représentent l'opérateur «ou inclusif» logique.

Démonstration 4.2
Soit Υ un ensemble fini de n séquences d'enchaînement x_i et x une séquence d'enchaînement de l'ensemble, si nous cherchons cette séquence d'enchaînement x dans l'ensemble Υ, nous pouvons dire que x est[5] soit la première[6] séquence d'enchaînement x_1, soit la deuxième x_2 ... soit la $n^{ième}$ x_n. Comme l'ensemble Υ est fini, d'après la définition 4.3, il est possible d'énumérer les séquences d'enchaînement de Υ, alors x est une des séquences d'enchaînement de l'ensemble et une seule séquence d'enchaînement de l'ensemble Υ.

Avec la propriété 4.3 et l'expression (4.3), il est possible d'introduire les mêmes définitions pour les séquences de précédences que pour les séquences d'enchaînement.

Définition 4.8 (Comparaison des séquences de précédences)
Les séquences de précédences sont considérées comme bien définies si nous possédons un critère permettant d'affirmer que deux de ces séquences de précédences, représentées par x_1 et par x_2, sont identiques *ou* distinctes *; nous écrirons :*

égalité : $x_1 = x_2$ *si et seulement si* toutes *les précédences de la séquence x_1 sont* identiques *à* toutes *les précédences de la séquence x_2 ;*

[5] mathématiquement égal

[6] Les appellations première, deuxième, $n^{ième}$ séquence d'enchaînement permettent de différencier les séquences d'enchaînement d'un ensemble et non de préciser leur position dans cet ensemble.

différence : $x_1 \neq x_2$ *si et seulement s'il y a* au moins une *précédence de la séquence* x_1 en contradiction *avec l'une des précédences de la séquence* x_2 *; c'est-à-dire par exemple si* $x_1 = \alpha_1 \rightarrow \alpha_2$ *et* $x_2 = \alpha_2 \rightarrow \alpha_1$.

Avec la définition 4.8, il est possible d'introduire la propriété suivante :

PROPRIÉTÉ 4.4 (FONCTION CARACTÉRISTIQUE DES SÉQUENCES DE PRÉCÉDENCES)
La fonction caractéristique $\mathcal{F}$ *d'un ensemble de séquences de précédences valides peut s'écrire :*

$$\mathcal{F}(x) = \sum_{i=1}^{n} \prod_{j=1}^{n-1} (x_{p_j} = (\alpha_{i,j} \rightarrow \alpha_{i,j+1})) \text{ avec } x = \prod_{j=1}^{n-1} (x_{p_j})$$

DÉMONSTRATION 4.3
Sachant que, quelque soit l'ensemble de séquences d'enchaînement, il existe une expression logique, exposée à la propriété 4.3, et que quelque soit la séquence d'enchaînement, il existe, d'après la propriété 4.1, la possibilité de transformer cette séquence d'enchaînement en une séquence de précédences, la propriété 4.4 est donc évidente.

Pour des raisons de simplicité de notation, nous supprimerons l'ensemble des comparaisons entre les variables[7] et les précédences de l'ensemble, en écrivant la propriété suivante :

PROPRIÉTÉ 4.5 (EXPRESSION LOGIQUE DES SÉQUENCES DE PRÉCÉDENCES)
L'expression logique $\pounds$ *d'un ensemble de séquences de précédences valides peut s'écrire :*

$$\pounds = \sum_{i=1}^{n} \prod_{j=1}^{n-1} (\alpha_{i,j} \rightarrow \alpha_{i,j+1})$$

REMARQUE 4.2
Les expressions de la forme $\sum_i \prod_j (\alpha_{i,j} \rightarrow \alpha_{i,j+1})$ *sont appelées expressions canoniques.*

4.2.4 Graphe de précédence

Un graphe de précédence représente un ensemble de contraintes de précédence, il est alors possible d'écrire la propriété suivante :

[7] Précédences de la séquence x.

Propriété 4.6 (Graphe de précédence)
Soit $G =< E, U >$ un graphe de précédence, où E est l'ensemble des tâches d'assemblage et U l'ensemble des contraintes de précédence et c_i sont des contraintes de précédence appartenant à U, le graphe de précédence G peut s'écrire :

$$G =< E, U > \Leftrightarrow \prod_{i=1}^{p} c_i$$

De là, il est possible d'en déduire l'écriture logique suivante d'un ensemble de n graphes de précédence G_i :

$$\{G_i =< E_i, U_i >\} \Leftrightarrow \sum_{i=1}^{n} \prod_{j=1}^{p} c_{ij} \tag{4.5}$$

4.2.5 Hypergraphe de précédence

De la même manière que pour les graphes de précédence il est possible de faire correspondre à un hypergraphe une forme logique :

Propriété 4.7 (Hypergraphe de précédence)
Soit $H =< X, \xi >$ un graphe de précédence, où X est l'ensemble des tâches d'assemblage et ξ l'ensemble des contraintes de précédence et les a_i représentent des contraintes de précédence c_i ou des disjonctions de contraintes de précédence d_{i_j} appartenant à ξ, l'hypergraphe de précédence H peut s'écrire :

$$H =< X, \xi > \Leftrightarrow \prod_{i=1}^{p} a_i$$

avec

$$a_i = c_i \ ou \ a_i = \sum d_{i_j}$$

De là, il est possible d'en déduire l'écriture logique suivante d'un ensemble de n hypergraphes de précédence H_i :

$$\{H_i =< X_i, \xi_i >\} \Leftrightarrow \sum_{i=1}^{n} \prod_{j=1}^{p} a_{ij} \tag{4.6}$$

Un exemple sera traité pour illustrer les quelques notions et définitions rappelées ci-avant.

4.2.6 Exemple

Considérons un exemple simple : soit un produit $\mathcal{P}$ constitué d'un ensemble de p composants élémentaires notés α, α_1, ..., α_{p-1}.

P est l'ensemble de $p-1$ tâches d'assemblage des constituants α_i de $\mathcal{P}$, de la tâche α de chargement du composant de base et de la tâche de déchargement u (4.7a). D est l'ensemble des séquences d'enchaînement possibles sur P. En l'absence de contraintes d'assemblage, le cardinal de D est A_{p-1}^{p-1} car le composant de base est fixé et la tâche de déchargement est toujours la dernière tâche de l'assemblage du produit considéré. Υ est l'ensemble fini des séquences d'enchaînement valides x_i (4.7b). Le complémentaire de Υ relatif à D est noté $\bar{\Upsilon}$ (4.7e), (4.7f). $\bar{\Upsilon}$ est l'ensemble fini des séquences d'enchaînement non valides y_i (4.7c). Chaque séquence d'enchaînement représente un ordre total sur P.

Le cardinal de Υ est n et le cardinal de $\bar{\Upsilon}$ est m avec $m = A_{p-1}^{p-1} - n$.

$$P = \{\alpha, \alpha_1, \ldots, \alpha_{p-1}, u\} \tag{4.7a}$$

$$\Upsilon = \{x_1, \ldots, x_n\} \tag{4.7b}$$

$$\bar{\Upsilon} = \{y_1, \ldots, y_m\} \tag{4.7c}$$

$$Card\, x_i = p+1 \quad \forall i \in \{1, \ldots, n\} \tag{4.7d}$$

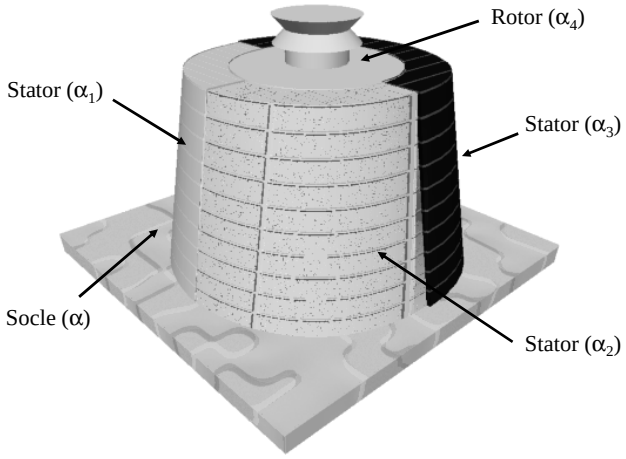
$$D = \Upsilon \cup \bar{\Upsilon} \tag{4.7e}$$

$$\Upsilon \cap \bar{\Upsilon} = \emptyset \tag{4.7f}$$

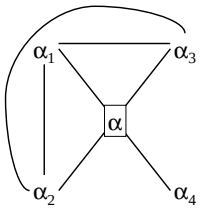
L'exemple utilisé ici, est l'assemblage d'un moteur schématique comportant cinq composants élémentaires. Les composants élémentaires de ce moteur, le socle α, les trois parties du stator α_1, α_2, α_3 et le rotor α_4 sont présentés dans la figure 4.1(a), et les liaisons entre ces constituants sont présentées par le graphe de liaisons de la figure 4.1(b).

Les figures 4.1(a) et 4.1(b) montrent que les séquences d'enchaînement de l'expression (4.9) vérifient la condition suivante :

> Le composant de base est le *socle* α, puis le *rotor* α_4 doit être posé avant les *stators* α_1 *et* α_2 ou avant le *stator* α_3 pour des raisons d'accessibilité. (4.8)



(a) Produit



(b) Graphe des liaisons

FIG. 4.1 – Moteur

$$
\begin{aligned}
x_1 &\equiv \alpha - \alpha_1 - \alpha_2 - \alpha_4 - \alpha_3 - u \\
x_2 &\equiv \alpha - \alpha_1 - \alpha_4 - \alpha_2 - \alpha_3 - u \\
x_3 &\equiv \alpha - \alpha_1 - \alpha_4 - \alpha_3 - \alpha_2 - u \\
x_4 &\equiv \alpha - \alpha_2 - \alpha_1 - \alpha_4 - \alpha_3 - u \\
x_5 &\equiv \alpha - \alpha_2 - \alpha_4 - \alpha_1 - \alpha_3 - u \\
x_6 &\equiv \alpha - \alpha_2 - \alpha_4 - \alpha_3 - \alpha_1 - u \\
x_7 &\equiv \alpha - \alpha_3 - \alpha_4 - \alpha_1 - \alpha_2 - u \\
x_8 &\equiv \alpha - \alpha_3 - \alpha_4 - \alpha_2 - \alpha_1 - u \\
x_9 &\equiv \alpha - \alpha_4 - \alpha_1 - \alpha_2 - \alpha_3 - u \\
x_{10} &\equiv \alpha - \alpha_4 - \alpha_2 - \alpha_1 - \alpha_3 - u \\
x_{11} &\equiv \alpha - \alpha_4 - \alpha_1 - \alpha_3 - \alpha_2 - u \\
x_{12} &\equiv \alpha - \alpha_4 - \alpha_2 - \alpha_3 - \alpha_1 - u \\
x_{13} &\equiv \alpha - \alpha_4 - \alpha_3 - \alpha_1 - \alpha_2 - u \\
x_{14} &\equiv \alpha - \alpha_4 - \alpha_3 - \alpha_2 - \alpha_1 - u
\end{aligned}
\tag{4.9}
$$

Avec la proposition 4.3 et avec l'ensemble Υ des séquences d'enchaînement de l'expression (4.9), nous pouvons écrire l'expression logique suivante :

$$\mathcal{L}_\Upsilon = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} \tag{4.10}$$

Dans nos travaux [49], nous avons introduit la notation logique $\pounds^*$ ((4.11) appelée $\pounds^*_\Upsilon$).

$$\pounds^*_\Upsilon = \bigoplus_{i=1}^{n} x_i = x_1 \oplus \ldots \oplus x_n \tag{4.11}$$

L'expression (4.11), inspirée de la proposition 4.3, traduit la recherche d'une séquence d'enchaînement parmi l'ensemble Υ des séquences d'enchaînement. La différence entre les deux expressions $\pounds$ et $\pounds^*$ est le type de relation existant entre chaque séquence d'enchaînement de l'ensemble Υ : dans la première expression, nous avons une relation *disjonctive inclusive*[8], tandis que dans la deuxième, cette relation est *disjonctive exclusive*[9]. Cependant, à l'aide de la démonstration 4.4, nous allons montrer l'équivalence entre $\pounds$ et $\pounds^*$, pour un ensemble Υ de séquences d'enchaînement.

$$\pounds_\Upsilon = \sum_{i=1}^{n} x_i = \bigoplus_{i=1}^{n} x_i \tag{4.12}$$

Démonstration 4.4
L'égalité (4.12) est effective car :

$$\pounds^*_\Upsilon = \bigoplus_{i=1}^{n} x_i = \sum_{i=1}^{n}((x_i \cdot \neg(\bigoplus_{j=1}^{n} x_j)) + (\bar{x_i} \cdot \bigoplus_{j=1}^{n} x_j)) \text{ avec } i \neq j \tag{4.13}$$

Appelons X_i le terme de la somme généralisée de l'expression 4.13.

$$X_i = (x_i \cdot \neg(\bigoplus_{j=1}^{n} x_j)) + (\bar{x_i} \cdot \bigoplus_{j=1}^{n} x_j) \text{ avec } i \neq j \tag{4.14}$$

En passant la négation à l'intérieur de la disjonction exclusive généralisée ($\oplus$) nous transformons l'expression (4.14) en l'expression (4.15).

$$X_i = (x_i \cdot (\prod_{j=1}^{n} \bar{x_j})) + (\bar{x_i} \cdot \bigoplus_{j=1}^{n} x_j) \text{ avec } i \neq j \tag{4.15}$$

Comme x_i appartient à l'ensemble Υ des séquences d'enchaînement, $\bar{x_i}$ n'appartient pas à cet ensemble, alors nous avons l'expression (4.16).

$$X_i = x_i \cdot \prod_{j=1}^{n} \bar{x_j} \text{ avec } i \neq j \tag{4.16}$$

[8] Opérateur «ou inclusif»
[9] Opérateur «ou exclusif»

En développant l'expression (4.16) et avec a_{i_k} les précédences de x_i, nous obtenons l'expression (4.17).

$$X_i = \prod_{k=1}^{p} a_{i_k} \cdot \prod_{j=1}^{n} (\sum_{k=1}^{p} a_{j_k}^{-}) \text{ avec } i \neq j \tag{4.17}$$

Comme les séquences d'enchaînement sont linéaires, toutes les tâches d'une séquence sont en relation de précédence deux à deux, nous avons alors, soit $\alpha_i \rightarrow \alpha_j$, soit $\alpha_j \rightarrow \alpha_i$. De plus comme une séquence est une permutation des tâches d'une autre séquence, nous pouvons écrire la propriété suivante :

Propriété 4.8 (Redondance des précédences)
Soit x_a et x_b deux séquences d'enchaînement, il existe deux tâches α_i et α_j telle que dans la séquence d'enchaînement x_a la tâche α_i précède la tâche α_j, et dans la séquence d'enchaînement x_b la tâche α_j précède la tâche α_i.
Cela peut aussi s'écrire :

$$\forall x_a, x_b \in \Upsilon, \exists \alpha_i, \alpha_j \in E | \alpha_i \rightarrow \alpha_j \in x_a, \alpha_j \rightarrow \alpha_i \in x_b$$

De l'équation (4.17) et de la propriété 4.8, nous pouvons déduire que quel que soit j, l'expression $\sum_{k=1}^{p} a_{j_k}^{-}$ est vraie. Nous avons alors :

$$X_i = \prod_{k=1}^{p} a_{i_k} \cdot \prod_{j=1}^{n} \text{ vrai } = \prod_{k=1}^{p} a_{i_k} = x_i \text{ avec } i \neq j \tag{4.18}$$

Après simplification de l'expression logique (4.4) à l'aide de l'expression (4.14), nous obtenons l'équation (4.19).

$$\pounds_{\Upsilon}^{*} = \sum_{i=1}^{n} x_i = \pounds_{\Upsilon} \tag{4.19}$$

4.3 Méthode logique proposée

La méthode mise en place ici, est la détermination des graphes de précédence à partir de l'ensemble des séquences d'enchaînement admissibles préalablement établies à l'aide de LEGA. Le principe de résolution est composé de cinq parties. La première partie est appelée le *développement logique*, il s'agit de la transformation d'un ensemble de

séquences d'enchaînement en un ensemble de relations logiques portant sur les précédences directes (voir définition 4.9). Une fois cet ensemble établi, la deuxième partie de la transformation se nomme *décomposition logique*. Elle est transcrite par la génération de toutes les précédences indirectes (voir définition 4.10). La troisième étape est la *réduction logique* de l'ensemble ainsi obtenu par factorisation. Cet ensemble factorisé est transformé en un ensemble de contraintes de précédences pouvant être assimilé à un ou plusieurs graphes de précédence ou hypergraphes de précédence, cette phase est appelé *génération des graphes*. Avant cette phase, nous devons introduire une étape de suppression des contraintes de précédence redondantes que nous appellerons *simplification logique*. Dans le meilleur des cas un graphe de précédence est obtenu, dans d'autres cas plus défavorables, un hypergraphe de précédence est engendré.

Définition 4.9 (Précédence directe)
Une précédence de la tâche a sur la tâche b est dite directe *si, dans la séquence d'enchaînement correspondante, il n'existe pas de tâche c entre les tâches a et b.*

Définition 4.10 (Précédence indirecte)
Une précédence de la tâche a sur la tâche b est dite indirecte *si, dans la séquence d'enchaînement correspondante, il existe une tâche c entre les tâches a et b.*

Dans l'exemple de la figure 4.1(a), avec la séquence d'enchaînement x_1, la tâche α précède *directement* la tâche α_1, mais *indirectement* la tâche α_2.

Les sections suivantes développent les cinq étapes de notre méthode de génération des graphes de précédence.

4.3.1 Décomposition logique

La *décomposition logique* $\mathcal{D}c$ est la transformation de l'ensemble des séquences d'enchaînement admissibles préalablement établies en une expression logique des précédences directes $\pounds_{\Upsilon}^{\mathcal{D}c}$.

Le processus de transformation est la génération d'une précédence directe pour chacun des enchaînements décrits dans chacune des séquences d'enchaînement. Toutes les précédences directes d'une séquence sont en relation à l'aide de l'opérateur de conjonction «$\cdot$» selon la propriété 4.1. Les sous-ensembles des précédences directes, générés par chaque séquence et appelés séquences de précédences sont mis en relation par l'opérateur disjonctif (noté «+») d'après la propriété 4.5.

Cette transformation est exprimée par l'algorithme 4.1. Comme la méthode proposée respecte l'équation (4.12), l'équivalence entre l'ensemble des séquences d'enchaînement et l'équation logique générée est établie.

Entrée : Ensemble de séquences d'enchaînement admissibles pré-établies avec LEGA

Sortie : Equation logique

Pour chaque séquence **faire**

// Génération d'une nouvelle disjonction de conjonction à chaque passage

Pour chaque précédence **faire**

Ajouter la précédence dans l'ensemble conjonctif

Fin pour

Ajouter la séquence de précédences générée dans l'ensemble disjonctif

Fin pour

ALGO 4.1 : Algorithme de décomposition logique d'un ensemble de séquences d'enchaînement en une équation logique

Cette transformation respecte les caractéristiques suivantes :

- Chaque séquence d'enchaînement génère p précédences directes.
- Le nombre N de précédences directes générées pour l'ensemble Υ des séquences d'enchaînement admissibles, est égal au nombre de séquences d'enchaînement admissibles «n» multiplié par le nombre de précédences générées par chaque séquence d'enchaînement «p», il est noté $N = n \times p$.

Cette décomposition logique est appliquée à l'exemple de la figure 4.1(a), pour donner l'expression (4.20).

Après la décomposition logique des séquences d'enchaînement en expression logique, nous passons au développement logique.

4.3.2 Développement logique

Le *développement logique* $\mathcal{D}v$ est un développement des précédences de l'expression logique $\pounds_{\Upsilon}^{\mathcal{D}c}$ par transitivité de l'opérateur de précédence en l'expression $\pounds_{\Upsilon}^{\mathcal{D}v}$. Cette *transitivité* de la précédence est exprimée par la propriété suivante :

$$\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}c} \equiv\ & (\alpha \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \\
& u)+(\alpha \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow u)+(\alpha \rightarrow \\
& \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow u)+(\alpha \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \\
& \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow u)+(\alpha \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \\
& \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow u)+(\alpha \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \\
& \alpha_1)\cdot(\alpha_1 \rightarrow u)+(\alpha \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \\
& u)+(\alpha \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow u)+(\alpha \rightarrow \\
& \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow u)+(\alpha \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \\
& \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow u)+(\alpha \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \\
& \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow u)+(\alpha \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \\
& \alpha_1)\cdot(\alpha_1 \rightarrow u)+(\alpha \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \\
& u) + (\alpha \rightarrow \alpha_4)\cdot(\alpha_4 \rightarrow \alpha_3)\cdot(\alpha_3 \rightarrow \alpha_2)\cdot(\alpha_2 \rightarrow \alpha_1)\cdot(\alpha_1 \rightarrow u)
\end{aligned} \tag{4.20}$$

Entrée : Equation logique

Sortie : Equation logique

Pour chaque ensemble conjonctif **faire**

Pour chaque paire possible de précédence **faire**

Si $(a \rightarrow b)$ et $(b \rightarrow c)$ **alors**

$(a \rightarrow c)$

Fin si

Fin pour

Fin pour

Algo 4.2 : Algorithme de développement logique d'un ensemble de précédences d'une équation logique

Propriété 4.9 (Transitivité de la précédence)
Si α_i précède α_j et α_j précède α_k alors α_i précède α_k.

$$\forall i,j,k|(\alpha_i \rightarrow \alpha_j)\cdot(\alpha_j \rightarrow \alpha_k) \Leftrightarrow (\alpha_i \rightarrow \alpha_j)\cdot(\alpha_j \rightarrow \alpha_k)\cdot(\alpha_i \rightarrow \alpha_k)$$

Ce développement permet de définir l'ensemble des précédences élargies entre toutes les tâches pour chacune des séquences d'enchaînement, en suivant le processus ci-après : pour chacune des séquences de précédences, ajouter les précédences indirectes déduites de toutes les paires de tâches possibles. Les précédences indirectes introduites sont re-

liées aux précédences directes de la séquence de précédences correspondante, c'est à dire que pour chaque sous-équation conjonctive, un ensemble de précédences indirectes est introduit.

La phase de développement logique est traduite par l'algorithme 4.2. Cet algorithme a pour donnée d'entrée l'ensemble des précédences directes, et l'ensemble des précédences élargies admissibles comme résultat, c'est-à-dire, l'ensemble de toutes les précédences directes et indirectes vérifiant l'intégrité de l'assemblage du produit.

Le développement logique introduit dans l'expression logique $p \times (p-1)/2$ précédences indirectes pour chaque séquence d'enchaînement, ce qui fait un apport de $n \times p \times (p-1)/2$ précédences indirectes. Après cette étape, l'expression logique comporte alors $n \times p^2/2$ précédences élargies[10].

Ce développement logique ne remet pas en cause l'équivalence entre l'ensemble des séquences d'enchaînement et l'équation logique obtenue, car la relation de précédence est transitive.

Si l'exemple de la figure 4.1(a) est traité avec cette méthode, le *développement logique* donne l'équation (4.21).

Après cette phase de développement, nous passons à la réduction logique de cette équation (4.21).

4.3.3 Réduction logique

La réduction logique $\mathcal{R}$ est la transformation d'une équation logique quelconque en une équation logique réduite $\pounds_{\Upsilon}^{\mathcal{R}}$. La réduction logique de notre expression revient à une simplification d'expressions booléennes.

Définition 4.11 (Équation logique réduite)
Nous appelons équation logique réduite*, toute équation logique sous sa forme minimale.*

Afin de réduire l'équation $\pounds_{\Upsilon}^{\mathcal{D}v}$, il existe plusieurs méthodes. Ici, seules trois d'entre elles sont présentées. Deux de ces méthodes, «*Quine-McCluskey*» et «*consensus*», sont connues par les automaticiens, tandis que la dernière est une méthode de réduction par factorisation, proposée par nous même.

[10] Nous ne les nommerons plus *précédences élargies* mais *précédences*.

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \\
& \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)
\end{aligned}
\tag{4.21}
$$

Prenons par exemple, l'expression logique de l'ensemble des séquences de précédences suivante :

$$\begin{gathered}\mathcal{L}_{\Upsilon}^{\mathcal{D}v}(\alpha,\alpha_1,\alpha_2,\alpha_3,u) = \\ \alpha\rightarrow\alpha_1\cdot\alpha_1\rightarrow\alpha_2\cdot\alpha_2\rightarrow\alpha_3\cdot\alpha_3\rightarrow u+ \\ \alpha\rightarrow\alpha_1\cdot\alpha_1\rightarrow\alpha_3\cdot\alpha_3\rightarrow\alpha_2\cdot\alpha_2\rightarrow u+ \\ \alpha\rightarrow\alpha_3\cdot\alpha_3\rightarrow\alpha_1\cdot\alpha_1\rightarrow\alpha_2\cdot\alpha_2\rightarrow u+ \\ \alpha\rightarrow\alpha_2\cdot\alpha_2\rightarrow\alpha_1\cdot\alpha_1\rightarrow\alpha_3\cdot\alpha_3\rightarrow u+ \\ \alpha\rightarrow\alpha_2\cdot\alpha_2\rightarrow\alpha_3\cdot\alpha_3\rightarrow\alpha_1\cdot\alpha_1\rightarrow u+ \\ \alpha\rightarrow\alpha_3\cdot\alpha_3\rightarrow\alpha_2\cdot\alpha_2\rightarrow\alpha_1\cdot\alpha_1\rightarrow u\end{gathered} \tag{4.22}$$

Il est possible d'effectuer les changements de variables suivants :

$$\begin{array}{llllll} \alpha\rightarrow\alpha_1 & : & x_1 & \alpha\rightarrow\alpha_2 & : & x_2 \\ \alpha\rightarrow\alpha_3 & : & x_3 & \alpha_1\rightarrow\alpha_2 & : & y \\ \alpha_2\rightarrow\alpha_3 & : & z & \alpha_1\rightarrow\alpha_3 & : & t \\ \alpha_1\rightarrow u & : & u_1 & \alpha_2\rightarrow u & : & u_2 \\ \alpha_3\rightarrow u & : & u_3 & \alpha\rightarrow u & : & x \end{array} \tag{4.23}$$

À l'aide des expressions (4.23) et de la propriété Enchaînement 4.2, nous pouvons écrire les changements de variables représentés par les expressions (4.24).

$$\begin{array}{llllll} \alpha_2\rightarrow\alpha_1 & : & \bar{y} & \alpha_3\rightarrow\alpha_2 & : & \bar{z} \\ \alpha_3\rightarrow\alpha_1 & : & \bar{t} & & & \end{array} \tag{4.24}$$

Par les changements de variables des expressions (4.23) et (4.24), nous transformons l'expression booléenne (4.22) en une expression booléenne (4.25).

$$\begin{gathered}\mathcal{L}_{\Upsilon}^{\mathcal{D}v}(x,x_1,x_2,x_3,y,z,t,u_1,u_2,u_3) = \\ x_1\cdot y\cdot z\cdot u_3+ \\ x_1\cdot\bar{z}\cdot t\cdot u_2+ \\ x_3\cdot y\cdot\bar{t}\cdot u_2+ \\ x_2\cdot\bar{y}\cdot t\cdot u_3+ \\ x_2\cdot z\cdot t\cdot u_1+ \\ x_3\cdot\bar{y}\cdot\bar{z}\cdot u_1\end{gathered} \tag{4.25}$$

4.3.3.1 Quine-McCluskey

La méthode de Quine-McCluskey est une méthode systématique basée sur la donnée de la forme disjonctive de la fonction à simplifier. L'algorithme consiste à regrouper

progressivement les *monômes*[11] *canoniques*[12] de degré n pour en faire des monômes de degré $n-1$ puis regrouper progressivement les monômes canoniques de degré $n-1$ pour en faire des monômes de degré $n-2$ et ainsi de suite. La tâche est réitérée jusqu'à ce qu'il n'y ait plus de regroupement possible. Les monômes qui n'interviennent dans aucun regroupement sont alors premiers. Comme deux monômes de degré n engendrent un monôme de degré $n-1$ si et seulement si ils diffèrent d'une seule composante, on commence par regrouper les monômes en classes, selon le nombre de variables niées qu'ils contiennent.

L'algorithme 4.3 peut être représenté par un tableau, comme le montre le tableau (4.28), où la première colonne est constituée de monômes canoniques que l'on regroupe par classes. Chacune des autres colonnes est obtenue à partir de celle qui la précède en regroupant les monômes appartenant à deux classes voisines.

Si nous utilisons l'expression (4.25), par transitivité nous avons alors :

$$\begin{array}{lll}
x_1 \cdot y \Rightarrow x_2 & y \cdot z \Rightarrow t & x_2 \cdot z \Rightarrow x_3 \\
x_3 \cdot u_3 \Rightarrow x & z \cdot u_3 \Rightarrow u_1 & t \cdot u_3 \Rightarrow u_1 \\
x_1 \cdot t \Rightarrow x_3 & t \cdot \bar{z} \Rightarrow y & x_3 \cdot \bar{z} \Rightarrow x_2 \\
x_2 \cdot u_2 \Rightarrow x & y \cdot u_2 \Rightarrow u_1 & \bar{z} \cdot u_2 \Rightarrow u_3 \\
x_3 \cdot \bar{t} \Rightarrow x_1 & y \cdot \bar{t} \Rightarrow \bar{z} & x_2 \cdot \bar{y} \Rightarrow x_1 \\
\bar{y} \cdot t \Rightarrow z & z \cdot \bar{t} \Rightarrow y & \bar{t} \cdot u_1 \Rightarrow u_3 \\
x_1 \cdot u_1 \Rightarrow x & \bar{y} \cdot u_1 \Rightarrow u_2 & \bar{y} \cdot \bar{z} \Rightarrow \bar{t}
\end{array} \tag{4.26}$$

Avec les expressions (4.25) et (4.26) nous pouvons écrire l'expression (4.27), qui est une fonction logique (par changement de variables).

$$\begin{array}{l}
\pounds_{\Upsilon}^{\mathcal{D}v}(x_1, x_2, x_3, x, y, z, t, u_1, u_2, u_3) = \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3 + \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot t \cdot u_1 \cdot u_2 \cdot u_3 + \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3 + \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3 + \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot z \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3 + \\
x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3
\end{array} \tag{4.27}$$

[11] Ensemble conjonctif de variables.

[12] Monômes possédant la totalité des variables affirmées ou niées intervenant dans l'expression.

En suivant l'algorithme, nous avons regroupé les monômes en fonction du nombre de variables niées puis nous avons engendré un monôme de degré inférieur.

$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3$
$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot t \cdot u_1 \cdot u_2 \cdot u_3$
$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot t \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot z \cdot u_1 \cdot u_2 \cdot u_3$
$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot u_1 \cdot u_2 \cdot u_3$
$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot z \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{z} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3$
$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3$	$x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3$

(4.28)

Pour cet exemple, la simplification ne peut pas se poursuivre, car les monômes ont tous, deux à deux, au moins deux variables affirmées ou niées différemment, ou ils n'ont pas les mêmes variables. Nous avons donc la base première.

Cette méthode, parce qu'elle nécessite la présence de toutes les variables dans les monômes c'est à dire la forme canonique[13] de la fonction, est relativement complexe à mettre en place et de plus, elle ne donne pas toujours la forme minimale de l'expression (voir tableau (4.28)). À partir de là nous devons, avec la table de choix [36], définir les monômes premiers suivants :

$$\begin{aligned} &x_1 \cdot x_2 \cdot x_3 \cdot x \cdot z \cdot t \cdot u_1 \cdot u_2 \cdot u_3 \\ &x_1 \cdot x_2 \cdot x_3 \cdot x \cdot \bar{y} \cdot \bar{t} \cdot u_1 \cdot u_2 \cdot u_3 \\ &x_1 \cdot x_2 \cdot x_3 \cdot x \cdot y \cdot \bar{z} \cdot u_1 \cdot u_2 \cdot u_3 \end{aligned} \tag{4.29}$$

4.3.3.2 Consensus

La méthode du consensus est une méthode systématique basée sur l'opération de consensus entre monômes. Elle s'applique aux fonctions représentées sous une forme polynômiale arbitraire.

Définition 4.12 (Consensus)
On dit que deux monômes m_1 et m_2 forment une paire productive lorsqu'une seule des variables apparaissant dans les deux monômes est biforme, c'est-à-dire affirmée dans l'un et niée dans l'autre. Dans ce cas, on appelle consensus de m_1 et m_2 le monôme noté $cons(m_1, m_2)$ qui est le produit des variables monoformes de m_1 et m_2.

Deux cas de figure se présentent :

[13] Forme disjonctive de conjonctions ($\sum \prod$)

Entrée : Ensemble de monômes canoniques

Sortie : Ensemble de monômes simplifiés

Répéter

Pour chaque paire de monômes de degré n de deux classes voisines **faire**

Si les deux monômes de cette paire diffèrent[14] d'une variable **alors**

Regrouper cette paire pour générer un monôme de degré $n-1$

Fin si

Fin pour

Les monômes non utilisés au cours de ce processus sont ajoutés au résultat

Jusqu'à non génération de monôme de degré $n-1$ ou génération d'un monôme de degré 0.

ALGO 4.3 : Algorithme de simplification d'une équation logique par la méthode de Quine-McCluskey

- $m_1 = xP$ et $m_2 = \bar{x}P$: $cons(m_1, m_2) = P$ est un nouveau monôme supérieur à m_1 et à m_2 qui peuvent ainsi s'éliminer ;
- $m_1 = xP$ et $m_2 = \bar{x}Q$: m_1 et m_2 mettent en commun leurs sommets pour engendrer un nouveau monôme par $cons(m_1, m_2)$; un cas favorable se produit lorsque m_1 ou m_2 est majoré par $cons(m_1, m_2)$.

Soit f une fonction représentée par une liste de monômes.

L'algorithme 4.4 de simplification de fonction booléenne de L se déroule en répétant les deux actions suivantes :

1. éliminer de la liste les monômes qui sont couverts par d'autres ;
2. former les consensus de chacun des monômes de la liste avec ceux qui le suivent ; ajouter à la liste les consensus qui ne sont pas couverts par un monôme existant ; aller à 1.

jusqu'à ce qu'il n'y ait plus de nouveaux monômes créés par consensus.

En traitant l'exemple de la fonction logique (4.22), nous trouvons l'expression (4.29) p. 108.

Entrée Sortie : L Equation logique
  Pour chaque monôme m du premier à l'avant dernier de L **faire**
    Pour chaque monôme p, du successeur de m au dernier de L **faire**
      Ajouter (consensus (m, p)) à L après (rang (m))
    Fin pour
  Fin pour

ALGO 4.4 : Algorithme de la méthode du consensus

Entrée : m monôme
  k rang
Entrée Sortie : L Equation logique
Variable : j entier
  $j \leftarrow k$
  Tant que $j <$ longueur$(L) + 1$ **faire**
    Si $c < L[j]$ **alors**
      retour
    Fin si
    Si $c > L[j] - 1$ **alors**
      supprimer $L[j]$
    Sinon
      incrémenter j
    Fin si
  Fin tant que
  $L \leftarrow L.\{c\}$

ALGO 4.5 : Algorithme d'ajout du consensus

La liste ainsi obtenue est celle des monômes premiers. Cette méthode ne donne pas toujours la base minimale.

REMARQUE 4.3
Ces deux méthodes permettent de réduire des fonctions logiques, mais ne conviennent pas dans notre cas très particulier où certaines variables sont des combinaisons logiques

d'autres variables. Elles ne permettent pas d'engendrer des graphes de précédence maximaux dans tous les cas. Cette non génération des graphes de précédence maximaux est due à l'objectif de ces deux méthodes qui est de définir la base minimale d'une fonction logique booléenne, tandis que nous nous cherchons à obtenir la forme canonique minimale[15] *de cette fonction.*

Le graphe de précédence associé à l'ensemble des séquences de précédence de l'équation (4.22) peut être représenté par l'expression (4.30). Car la variable x peut être déduite de l'ensemble des autres variables, et pour les variables y, z, t, l'une d'entre elles dépend toujours des deux autres (voir équation (4.26) p. 107).

$$\pounds_{\Upsilon}^{\mathcal{R}}(x_1, x_2, x_3, x, y, z, t, u_1, u_2, u_3) = x_1 \cdot x_2 \cdot x_3 \cdot u_1 \cdot u_2 \cdot u_3 \tag{4.30}$$

4.3.3.3 Méthode de réduction proposée

La troisième étape est la *réduction logique* de l'équation logique $\mathcal{R}$, à l'aide de la *factorisation*, qui nous donne l'expression $\pounds_{\Upsilon}^{\mathcal{R}}$. Cette méthode s'applique sur des expressions polynômiales arbitraires, cependant, il est préférable de l'appliquer sur des monômes canoniques, car les précédences représentées par ces monômes sont liées entre elles par transitivité. C'est pour cette raison que la méthode proposée comporte la phase *développement logique*.

La *factorisation* est la factorisation logique classique, $(((\alpha_i \rightarrow \alpha_j) \cdot (\alpha_k \rightarrow \alpha_l)) + ((\alpha_i \rightarrow \alpha_j) \cdot (\alpha_m \rightarrow \alpha_n))) \Leftrightarrow (\alpha_i \rightarrow \alpha_j) \cdot ((\alpha_k \rightarrow \alpha_l) + (\alpha_m \rightarrow \alpha_n))$ ou $((\alpha_i \rightarrow \alpha_j) \cdot (\alpha_k \rightarrow \alpha_l)) + ((\alpha_m \rightarrow \alpha_n) \cdot (\alpha_o \rightarrow \alpha_p)) \Leftrightarrow ((\alpha_i \rightarrow \alpha_j) + (\alpha_m \rightarrow \alpha_n)) \cdot ((\alpha_k \rightarrow \alpha_l) + (\alpha_m \rightarrow \alpha_n)) \cdot ((\alpha_i \rightarrow \alpha_j) + (\alpha_o \rightarrow \alpha_p)) \cdot ((\alpha_k \rightarrow \alpha_l) + (\alpha_o \rightarrow \alpha_p))$

En ce qui concerne la *factorisation*, nous nous basons sur l'appellation *2-SAT*(*2-satisfability*), que K.S. Naphade [43] a rappelée (voir Annexe B), pour introduire une nouvelle notion : la *k-stabilité*[16] ou *k-STA*, avec k entier strictement positif.

Définition 4.13 (k-STA)
Clause contenant des opérateurs «·» de conjonction entre chaque prédicat, ces prédicats pouvant contenir au maximum $k-1$ opérateurs disjonctifs.

[15] Forme possédant le minimum d'opérateurs de disjonction et de conjonction.

[16] Nous appelons *stabilité* la réalisation d'un type de factorisation ; soit de simple niveau, soit de niveau deux.

Le choix du terme «*stabilité*» a été orienté par la sensation de stabilité de l'expression logique lors de la factorisation, c'est à dire que cette expression est dans sa forme minimale disjonctive de conjonctions de contraintes de précédence ($\sum\prod$) pour la forme 1-STA et dans sa forme minimale disjonctive de conjonctions pouvant contenir des disjonctions des contraintes de précédence ($\sum\prod\sum$) pour la forme 2-STA.

La transformation de l'ensemble des disjonctions de conjonctions de contraintes de précédence a atteint :

- la *1-stabilité* ou *1-STA* lorsque l'ensemble résultat est un ensemble disjonctif de conjonctions de contraintes de précédence, que nous notons : $\sum\prod(\alpha_i \rightarrow \alpha_j)$, où chaque disjonction est indépendante des autres,
- la *2-stabilité* ou *2-STA* lorsque l'ensemble résultat est un ensemble disjonctif de conjonctions de disjonctions de contraintes de précédence, que nous notons : $\sum\prod\sum(\alpha_i \rightarrow \alpha_j)$,

avec $i \neq j$ et $i \in \{1, \ldots, p\}$ et $j \in \{1, \ldots, p\}$.

Ce qui peut s'illustrer par les expressions suivantes pour les stabilités :
1-STA : $(\alpha_i \rightarrow \alpha_j) \cdot (\alpha_k \rightarrow \alpha_l) + (\alpha_i \rightarrow \alpha_j) \cdot (\alpha_k \rightarrow \alpha_j)$
2-STA : $(\alpha_i \rightarrow \alpha_j) \cdot ((\alpha_k \rightarrow \alpha_l) + (\alpha_m \rightarrow \alpha_l))$

La réduction logique est traduite par l'algorithme 4.6, qui a pour données d'entrée, l'expression logique développée $\pounds_{\Upsilon}^{\mathcal{D}^v}$ d'assemblage d'un produit, et pour résultat, une expression logique $\pounds_{\Upsilon}^{\mathcal{R}}$ factorisée. Nous mettons ainsi en facteur toutes les précédences qui peuvent l'être, puis nous découpons le problème en deux sous-problèmes distincts, et nous recommençons avec ces deux parties, et ce jusqu'à ce qu'il ne reste plus de précédence non traitée. Sachant que cette factorisation respecte toutes les règles et propriétés de la factorisation d'équation logique de l'algèbre booléenne. Nous avons alors l'équivalence entre l'ensemble des séquences d'enchaînement et l'équation logique factorisée, $\pounds_{\Upsilon}^{\mathcal{R}}$.

La réduction logique appliquée sur notre exemple de la figure 4.1(a) donne en *1-STA* (les différentes étapes entre l'équation 4.21 et l'équation 4.31 sont proposées en annexe C) :

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{R}_{1-STA}} \equiv ((\alpha \rightarrow \alpha_1) \cdot (\alpha \rightarrow \alpha_2) \cdot (\alpha \rightarrow \alpha_3) \cdot (\alpha \rightarrow \alpha_4) \cdot (\alpha \rightarrow \\ u) \cdot (\alpha_1 \rightarrow u) \cdot (\alpha_2 \rightarrow u) \cdot (\alpha_3 \rightarrow u) \cdot (\alpha_4 \rightarrow u) \cdot (\alpha_4 \rightarrow \alpha_3)) + ((\alpha \rightarrow \\ \alpha_1) \cdot (\alpha \rightarrow \alpha_2) \cdot (\alpha \rightarrow \alpha_3) \cdot (\alpha \rightarrow \alpha_4) \cdot (\alpha \rightarrow u) \cdot (\alpha_1 \rightarrow u) \cdot (\alpha_2 \rightarrow \\ u) \cdot (\alpha_3 \rightarrow u) \cdot (\alpha_3 \rightarrow \alpha_4) \cdot (\alpha_4 \rightarrow u) \cdot (\alpha_4 \rightarrow \alpha_1) \cdot (\alpha_4 \rightarrow \alpha_2))\end{aligned} \tag{4.31}$$

La réduction logique appliquée sur notre exemple de la figure 4.1(a) donne en *2-STA* (les différentes étapes entre l'équation 4.21 et l'équation 4.32 sont proposées en an-

Entrée : Equation logique

Sortie : Equation logique factorisée

Tant que il existe une précédence commune dans toutes les séquences **faire**

Mettre en facteur cette précédence commune

Fin tant que

Si il existe une disjonction de précédences commune à plusieurs séquences **alors**

Mettre en facteur la disjonction de précédences commune au plus grand nombre de séquences
Relancer l'algorithme avec les séquences factorisées
Relancer l'algorithme avec le reste des séquences

Sinon

Retourner le résultat

Fin si

ALGO 4.6 : Algorithme de réduction d'une équation logique

nexe D) :

$$\pounds_{\Upsilon}^{\mathcal{R}_{2-STA}} \equiv (\alpha \rightarrow \alpha_1) \cdot (\alpha \rightarrow \alpha_2) \cdot (\alpha \rightarrow \alpha_3) \cdot (\alpha \rightarrow \alpha_4) \cdot (\alpha \rightarrow u) \cdot (\alpha_1 \rightarrow u) \cdot (\alpha_2 \rightarrow u) \cdot (\alpha_3 \rightarrow u) \cdot (\alpha_4 \rightarrow u) \cdot ((\alpha_4 \rightarrow \alpha_1) + (\alpha_4 \rightarrow \alpha_3)) \cdot ((\alpha_4 \rightarrow \alpha_2) + (\alpha_4 \rightarrow \alpha_3)) \tag{4.32}$$

4.3.4 Simplification logique

La *simplification logique* est la suppression des contraintes de précédence indirectes, c'est-à-dire qui peuvent être déduites des autres contraintes de précédence. Pour cela nous recherchons toutes les conjonctions de trois contraintes de précédence du type : $(\alpha_i \rightarrow \alpha_j) \cdot (\alpha_j \rightarrow \alpha_k) \cdot (\alpha_i \rightarrow \alpha_k)$ afin de les réduire à l'expression :$(\alpha_i \rightarrow \alpha_j) \cdot (\alpha_j \rightarrow \alpha_k)$.

Cette simplification est exprimée par l'algorithme 4.7. La simplification logique respecte la suppression des termes redondants sans supprimer de contraintes non encore exprimées. Elle permet de garder l'équivalence entre l'ensemble des séquences d'enchaînement et l'équation ainsi obtenue.

Entrée : Equation logique

Sortie : Equation logique simplifiée

Répéter

Pour chaque ensemble de conjonctions **faire**

Pour chaque sous-ensemble possible de trois contraintes de précédence de type $(a \rightarrow b)$ et $(b \rightarrow c)$ et $(a \rightarrow c)$ **faire**

Supprimer la contrainte de précédence $(a \rightarrow c)$.

Fin pour

Fin pour

Pour chaque disjonctions **faire**

Si $(a \rightarrow b) + (b \rightarrow a)$ **alors**

Supprimer la disjonction des deux précédences

Fin si

Fin pour

Jusqu'à stabilité.

ALGO 4.7 : Algorithme de simplification logique d'une équation logique

La simplification logique appliquée sur notre exemple en *1-STA* de la figure 4.1(a) donne l'équation (4.33), notée $\pounds_{\Upsilon}^{1-STA}$.

$$\begin{aligned}\pounds_{\Upsilon}^{1-STA} \equiv ((\alpha \rightarrow \alpha_1) \cdot (\alpha \rightarrow \alpha_2) \cdot (\alpha \rightarrow \alpha_4) \cdot (\alpha_1 \rightarrow u) \cdot (\alpha_2 \rightarrow \\ u) \cdot (\alpha_3 \rightarrow u) \cdot (\alpha_4 \rightarrow \alpha_3)) + ((\alpha \rightarrow \alpha_3) \cdot (\alpha_3 \rightarrow \alpha_4) \cdot (\alpha_1 \rightarrow u) \cdot (\alpha_2 \rightarrow \\ u) \cdot (\alpha_4 \rightarrow \alpha_1) \cdot (\alpha_4 \rightarrow \alpha_2))\end{aligned} \tag{4.33}$$

La simplification logique appliquée sur notre exemple en *2-STA* de la figure 4.1(a) donne l'équation (4.34), notée $\pounds_{\Upsilon}^{2-STA}$.

$$\begin{aligned}\pounds_{\Upsilon}^{2-STA} \equiv (\alpha \rightarrow \alpha_1) \cdot (\alpha \rightarrow \alpha_2) \cdot (\alpha \rightarrow \alpha_3) \cdot (\alpha \rightarrow \alpha_4) \cdot (\alpha_1 \rightarrow \\ u) \cdot (\alpha_2 \rightarrow u) \cdot (\alpha_3 \rightarrow u) \cdot ((\alpha_4 \rightarrow \alpha_1) + (\alpha_4 \rightarrow \alpha_3)) \cdot ((\alpha_4 \rightarrow \\ \alpha_2) + (\alpha_4 \rightarrow \alpha_3))\end{aligned} \tag{4.34}$$

Après la *simplification*, la transformation logique de l'écriture en un ou plusieurs graphes ou hypergraphes de précédence est développée dans la section suivante.

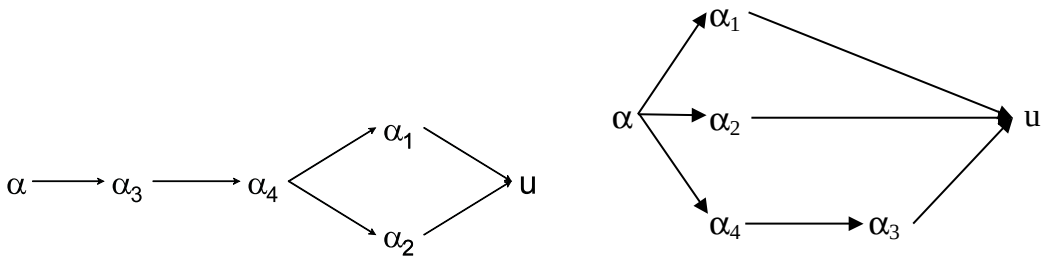
4.4 La génération des graphes

La génération des graphes de précédence est la transformation d'une écriture logique des contraintes de précédence en un ou plusieurs ensembles de contraintes de précédence. Selon le nombre et le type d'ensembles générés, nous obtenons un ou plusieurs graphes de précédence de types différents : graphe de précédence ou hypergraphe de précédence.

Pour obtenir un *graphe de précédence*, nous devons avoir l'ensemble E des constituants et l'ensemble U des contraintes de précédence directes. Pour obtenir un *hypergraphe de précédence*, nous devons avoir l'ensemble X des constituants et l'ensemble ξ des contraintes disjonctives ou non de précédence directes.

4.4.1 Les graphes de précédence

Afin de générer des graphes de précédence valides nous devons obtenir une équation logique de type *1-STA* comme nous avons pu le voir ci-avant. Une fois obtenu cet ensemble de disjonctions de conjonctions de contraintes de précédence, nous pouvons générer un ensemble de graphes de précédence à l'aide de l'algorithme 4.8. Cet algorithme génère, à l'aide de chaque ensemble conjonctif de contraintes de précédence, un graphe de précédence distinct. Pour cela, nous engendrons l'ensemble E des constituants, puis nous insérons chacune des contraintes de précédence dans l'ensemble U des contraintes de précédence.



(a) Graphe de précédence (b) Graphe de précédence

FIG. 4.2 – Graphes de précédence d'assemblage du moteur

Si nous prenons l'exemple de la figure 4.1(a), avec les séquences admissibles vérifiant la condition (4.8), l'ensemble des graphes de précédence peut s'écrire :

Entrée : Equation logique simplifiée
  Pour chaque ensemble de conjonctions **faire**
    // Génération d'un nouveau graphe de précédence à chaque passage
    Insérer la tâche de chargement du composant de base dans E
    Pour chaque précédence **faire**
      // Recherche de la tâche précédée dans la relation
      Si la tâche n'appartient pas à l'ensemble E **alors**
        Insérer la tâche dans E
      Fin si
      Insérer la contrainte de précédence dans U
    Fin pour
  Fin pour
Sortie : Ensemble de graphes de précédence

ALGO 4.8 : Algorithme de génération d'un ensemble de graphes de précédence

$\{G_1 =< E_1, U_1 >, G_2 =< E_2, U_2 >\}$ avec :
$E_1 = \{\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4, u\}$
$U_1 = \{(\alpha, \alpha_3), (\alpha_4, \alpha_1), (\alpha_4, \alpha_2), (\alpha_3, \alpha_4), (\alpha_1, u), (\alpha_2, u)\}$
et
$E_2 = \{\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4, u\}$
$U_2 = \{(\alpha, \alpha_1), (\alpha, \alpha_2), (\alpha, \alpha_4), (\alpha_4, \alpha_3), (\alpha_1, u), (\alpha_2, u), (\alpha_3, u), \}.$

Les deux graphes de précédence $G_1 =< E_1, U_1 >, G_2 =< E_2, U_2 >$ ci-avant du produit de la figure 4.1(a) p. 98 sont représentés respectivement par les figures 4.2(a) et 4.2(b).

4.4.2 Les hypergraphes de précédence

Un ensemble de conjonctions de disjonctions de contraintes de précédence de type *2-STA* permet d'obtenir un hypergraphe de précédence. Avec l'algorithme 4.9 et l'exemple de la figure 4.1(a), nous générons l'expression ci-après représentant l'hypergraphe de précédence H qui est représenté graphiquement dans la figure 4.3.

Un hypergraphe de précédence est équivalent à un ensemble conjonctif de contraintes de précédence et de contraintes disjonctives de précédence ($\prod \sum$). Cet algorithme génère un nouvel hypergraphe à chaque ensemble conjonctif de disjonctions de contraintes de précédence. Et si nous ne pouvons pas l'écrire sous forme de conjonctions de disjonctions de contraintes de précédence alors nous écrivons cette équation sous sa forme disjonctive de conjonctions de disjonctions de précédence ($\sum \prod \sum$), et grâce à cette transformation nous arrivons à représenter cet ensemble par un ensemble d'hypergraphes de précédence.

Entrée : Equation logique

Pour chaque ensemble de conjonctions de disjonctions de contraintes de précédence **faire**

// Génération d'un nouvel hypergraphe de précédence à chaque passage

Insérer la tâche de chargement du composant de base dans E

Pour chaque précédence ou disjonction de contraintes de précédence **faire**

// Affectation des contraintes de précédences dans l'hypergraphe considéré

Si les tâches n'appartiennent pas à l'ensemble X **alors**

Insérer les tâches dans X

Fin si

Insérer la contrainte de précédence ou la disjonction de contraintes de précédence dans ξ

Fin pour

Fin pour

Sortie : Ensemble d'hypergraphes de précédence

ALGO 4.9 : Algorithme de génération d'un ensemble d'hypergraphes de précédence

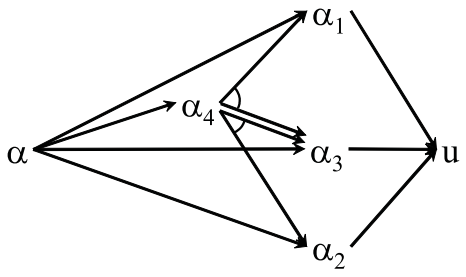


FIG. 4.3 – Hypergraphe de précédence d'assemblage du moteur

La figure 4.3 représente l'hypergraphe de précédence du produit de la figure 4.1(a), soit formellement :

$H = (X, \xi)$ avec :
$X = \{\alpha, \alpha_1, \alpha_2, \alpha_3, \alpha_4, u\}$
$\xi = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9\}$
$E_1 = (\{\alpha\}, \{\alpha_1\}), E_2 = (\{\alpha\}, \{\alpha_2\}), E_3 = (\{\alpha\}, \{\alpha_3\}),$
$E_4 = (\{\alpha\}, \{\alpha_4\}), E_5 = (\{\alpha_4\}, \{\alpha_1, \alpha_3\}), E_6 = (\{\alpha_4\}, \{\alpha_2, \alpha_3\}),$
$E_7 = (\{\alpha_1\}, \{u\}), E_8 = (\{\alpha_2\}, \{u\}), E_9 = (\{\alpha_3\}, \{u\}).$

4.5 Complexité

Dans cette section nous allons étudier la complexité de notre méthode, cette étude permettra de maximiser le nombre de contraintes de précédence intervenant dans l'expression logique, ainsi que d'évaluer le nombre de tâches à effectuer pour chacune des phases.

Pour ces calculs de complexité, nous prendrons deux variables, p le nombre de constituants à assembler et n le nombre de séquences d'enchaînement dans l'ensemble Υ.

Comme nous l'avons vu au long de ce chapitre, la transformation de l'ensemble Υ des n séquences d'enchaînement admissibles des $p+1$ tâches en un ensemble de précédences directes, est appelée *décomposition logique*. Cette décomposition logique est le découpage de chacune des n séquences en p précédences directes, ce qui donne $n \times p$ précédences directes. Pour cela l'algorithme 4.1 effectue $n \times p$ tâches du type génération d'une précédence ou alors affectation d'une précédence à un ensemble par exemple. La complexité de l'algorithme de décomposition est en $O(n \times p)$.

Pour la deuxième phase, le *développement logique* transforme l'expression, obtenue ci-avant avec la décomposition logique, en lui ajoutant un ensemble de termes, appelés

précédences indirectes. Le nombre de ces précédences indirectes est fonction du nombre p de précédences directes de chacune des séquences et du nombre n de séquences. Pour chacune des séquences, il est généré $p \times (p-1)/2$ précédences indirectes. L'expression logique résultant de ce développement logique comporte alors $n \times p \times (p-1)/2$ précédences indirectes ajoutées aux $n \times p$ précédences directes déjà présentes, ce qui donne $n \times p^2/2$ précédences. L'algorithme a une complexité en $O(n \times (p^2-1)/2)$.

En ce qui concerne la réduction logique, il est possible de dire qu'un nombre minimal de précédences est mis en facteur, comme :

- la tâche de chargement du composant de base précède toutes les autres tâches (p) ;
- toutes les tâches précèdent la tâche de déchargement du produit fini (p).

Cependant la précédence de la tâche de chargement du composant de base sur la tâche de déchargement du produit fini appartient aux deux ensembles, alors cela nous donne $2p-1$ précédences au minimum. Cela nous conduit à $n \times p^2/2 - 2p - 1$ précédences aux maximum après cette étape. L'algorithme 4.6 réduit cette expression logique en $n \times p$ cycles. L'algorithme de réduction logique a une complexité en $O(p^2/2)$.

La quatrième phase est la *simplification*, elle permet de supprimer l'ensemble des contraintes de précédences indirectes. Il est difficile de donner le nombre de contraintes de précédences indirectes éliminées car, selon les produits à assembler, ce nombre peut varier de zéro à $n \times p^2/2 - 2p - 1$ moins le nombre de précédences indispensables $(2p-3)$, ce qui donne $n \times p^2/2 - 4p - 4$. Avec la théorie mathématique, l'algorithme de simplification a une complexité maximale en $O((n \times p^2)!)$, cependant il est facile de réduire la complexité, en triant les contraintes de précédence en fonction de la première tâche et en sélectionnant la tâche intéressante à l'aide d'un algorithme de sélection approprié, en $O(p^3)$.

La cinquième et dernière étape est la *génération des graphes*, par la transformation d'une expression logique en un ou plusieurs ensembles de contraintes de précédence et en un ou plusieurs ensembles de constituants. Cette phase a une complexité maximale en $O(n \times p)$.

D'après cette analyse, la complexité de la méthode est en $O(p^3 + n \times p^2)$. Nous voyons que celle-ci est d'ordre polynômial.

4.6 Conclusion du chapitre

Ce chapitre propose une nouvelle méthode de détermination des graphes de précédence ou des hypergraphes de précédence d'assemblage à partir d'ensembles de séquences

d'enchaînement préalablement déterminées. Ces séquences d'enchaînement doivent satisfaire les hypothèses introduites au chapitre 2 p. 30.

Cette nouvelle approche de la génération des graphes de précédence par la logique booléenne est composée de quatre phases. La première d'entre elles, appelée *décomposition logique*, permet de transformer une séquence d'enchaînement en une séquence de précédences, où les précédences sont mises en relation à l'aide de l'opérateur de conjonction «·». Après la transformation de toutes les séquences d'enchaînement en séquences de précédences, ces dernières sont mises en relation avec l'opérateur disjonctif «+». L'*expression logique d'un ensemble* détermine la mise en relation des séquences de précédences.

L'expression, ainsi obtenue, est transformée par la deuxième phase appelée *développement logique*. Cette étape permet d'introduire l'ensemble des précédences indirectes de chacune des séquences de précédences. Les précédences indirectes de chacune des séquences de précédences, sont mises en relation avec les précédences directes par l'opérateur conjonctif.

Après cette phase, la troisième étape est appelée *réduction logique*. Dans ces travaux, trois méthodes de réduction sont exposées.

La première est la méthode de Quine-McCluskey. Seules les fonctions logiques sous *forme canonique*[17] peuvent être réduites. Cette méthode est pertinente et bien connue des automaticiens, mais ne garantit pas l'obtention de la forme minimale, qui permettrait de réaliser un hypergraphe de précédence. De plus cette méthode ne nous permet pas de générer facilement des graphes de précédence maximaux.

La deuxième méthode proposée est la méthode du consensus. Elle est aussi connue que la méthode de Quine-McCluskey. C'est une méthode de détermination d'une forme réduite, mais ne donnant pas toujours la forme minimale.

Pour résoudre ce problème, nous avons introduit notre propre méthode, qui est une factorisation des variables, selon qu'elles sont affirmées ou niées dans les séquences de précédences. Il y a plusieurs stades de factorisation, nous les appelons k-stabilité (k-STA). La 1-STA est l'état atteint après factorisation de type simple, c'est à dire que nous restons sous une forme disjonctive de conjonctions de contraintes de précédence ($\sum\prod$). Cet état permet d'engendrer des graphes de précédence uniquement. Si nous voulons des hypergraphes de précédence, nous devons atteindre la 2-STA, c'est à dire factoriser l'expression pour obtenir une forme disjonctive de conjonctions contenant elles mêmes des disjonctions de contraintes de précédence. Dans les cas les plus favorables,

[17] Tous les monômes sont canoniques

nous n'atteignons pas la 2-STA, car l'ensemble des séquences d'enchaînement peut être représenté par un graphe de précédence unique.

Pour obtenir des graphes de précédence ou hypergraphes de précédence ne possédant pas de redondance d'information et pour les alléger, une quatrième étape a été introduite, elle s'appelle *simplification logique*. Cette phase est la suppression de toutes les contraintes de précédence pouvant être déduites par transitivité des autres contraintes de précédence.

Cette méthode de génération des graphes et hypergraphes de précédence peut être qualifiée de *simple* et *efficace*.

Conclusion et perspectives

AUJOURD'HUI, les systèmes de production, à cause de la concurrence, doivent garantir un service de qualité, tout en restant compétitifs. Cela est aussi valable pour les systèmes d'assemblage, qui sont à l'origine de ces travaux. Le *système d'assemblage optimum* est généralement obtenu par des méthodes d'équilibrage de charge de lignes d'assemblage (ALB)[18]. Ces méthodes sont principalement utilisées avec l'une des représentations des processus d'assemblage, les *graphes de précédence*. L'utilité de cette représentation est due principalement à sa lisibilité, sa compacité et sa flexibilité, mais elle possède un défaut important : sa génération. Jusqu'ici, nous jugions les méthodes de génération systématique des graphes de précédence peu satisfaisantes.

Ces travaux nous présentent un échantillon des représentations des processus d'assemblage possibles. Les deux méthodes les plus utilisées sont les *graphes de précédence* pour la conception de systèmes d'assemblage et les *réseaux de Petri* pour le pilotage de systèmes. En ce qui concerne le pilotage, il existe un grand nombre de travaux concrétisés par des simulateurs, tandis qu'en conception de systèmes d'assemblage, il existe des méthodes de découpage de graphes de précédence pour l'ALB, mais il n'existe pas de méthode simple et efficace de génération des graphes de précédence. Ces travaux ont donc pour objectif de proposer une méthode de génération systématique des graphes de précédence *simple* et *efficace*, à partir d'un ensemble de graphes d'assemblage préalablement établi à l'aide de LEGA. Nous avons présenté deux méthodes différentes, basées pour l'une, sur la transformation de graphes, et pour l'autre, sur la logique booléenne.

La première de ces deux méthodes est une suite de transformations de graphes. Ces graphes sont, successivement, un graphe d'état intermédiaire du produit (ASTD), le graphe dual de l'ASTD appelé *graphe d'enchaînement*, puis à l'aide de règles de suppression d'arcs, un graphe d'enchaînement simplifié est obtenu. Sous certaines conditions, le graphe ainsi généré est équivalent à un graphe de précédence. Sinon dans les autres

[18] Assembly Line Balancing

cas, nous pouvons générer un hypergraphe de précédence. Cette méthode est transcrite par un ensemble d'algorithmes, dans ces travaux, elle permet d'obtenir des graphes de précédence valides et exhaustifs dans les cas où l'ensemble des séquences d'enchaînement proposé peut être représenté par un *graphes de précédence unique*. Dans ces cas, cette méthode conduit simplement et directement à un graphe de précédence.

Pour garantir l'obtention d'un graphe de précédence unique valide et exhaustif, nous devons vérifier que les séquences d'enchaînement de l'ensemble étudié vérifient la propriété Π proposée par A. Bratcu, puis enfin, appliquer notre méthode de génération systématique des graphes de précédence.

Pour obtenir des hypergraphes de précédence, nous devons utiliser la deuxième amélioration apportée, c'est à dire l'utilisation des précédences conditionnelles[19]. De là, il est relativement facile de générer des hyperarcs de précédence, un hypergraphe de précédence découle de tous les hyperarcs générés.

Les calculs de complexité des différents algorithmes permettent de dire que cette méthode a une complexité d'ordre polynômial, mais après le traitement d'un exemple, nous voyons que l'utilisation de celle-ci n'est pas triviale pour la génération d'hypergraphes. Il ressort de cette étude, que la méthode par transformation de graphes proposée est efficace, mais pas simple dans l'ensemble des cas, c'est pour cela que nous proposerons alors une seconde méthode.

Pour la seconde méthode, nous nous sommes inspirés des travaux de K.S. Naphade pour exprimer les contraintes de précédence sous forme d'expression logique booléenne. En définissant la fonction caractéristique de l'ensemble des séquences d'enchaînement, nous avons obtenu une expression logique booléenne, notée $\pounds$, cette étape est appelée *décomposition logique*.

Avec cette expression logique, nous générons un ensemble de graphes ou d'hypergraphes de précédence à l'aide des phases suivantes, le *développement logique des précédences*, la *réduction logique des précédences*, la *simplification logique d'une expression* et la *génération de graphes de précédence* ou la *génération d'hypergraphes de précédence*.

Le *développement logique* est basé sur la transitivité de l'opérateur de précédence «→», il permet de transformer l'ensemble des précédences directes en un ensemble de précédences élargie[20].

[19] Cette notion de précédence conditionnelle permet d'exprimer la disjonction entre deux précédences.
[20] Précédences directes et indirectes

La *réduction logique des précédences* est une méthode de factorisation des précédences afin d'obtenir la base minimale représentative de l'ensemble des séquences d'enchaînement. Pour cette étape, nous avons introduit une nouvelle appellation, la k-stabilité[21] (k-STA). Cette appellation est inspirée de l'appellation 2-SAT de K.S. NAPHADE pour exprimer le degré de factorisation d'une expression logique, c'est à dire si l'expression logique peut s'écrire sous une forme de conjonctions de contraintes de précédence c_i ($\prod c_i$) ou sous une forme disjonctive de conjonctions de contraintes de précédence ($\sum \prod c_i$) alors k est égal à un (1-STA), sinon si cette expression peut s'écrire sous la forme conjonctive de contraintes de précédence ou de disjonctions de contraintes de précédence ($\prod \sum$) ou sous la forme disjonctive de conjonctions de contraintes de précédence ou de disjonctions de contraintes de précédence ($\sum \prod \sum$), alors k est égal à deux (2-STA).

Après cette étape de détermination de la base minimale, selon son degré de stabilité, la *simplification logique d'une expression* est la suppression de toutes les contraintes de précédences indirectes, afin d'obtenir l'ensemble minimal des contraintes de précédence.

Il est alors possible de générer un graphe de précédence, un ensemble de graphes de précédence, un hypergraphe de précédence ou alors un ensemble d'hypergraphes de précédence. De la base de stabilité «1-STA», la *génération d'un ensemble de graphes de précédence* permet de transformer un ensemble de contraintes de précédence en un graphe ou un ensemble de graphes de précédence. Il en est de même avec la *génération d'un ensemble d'hypergraphes de précédence* avec la base de stabilité «2-STA» pour les hypergraphes de précédence.

Cette méthode engendre *l'ensemble des graphes de précédence*, représentant de manière biunivoque l'ensemble des séquences d'enchaînement initial, elle permet aussi de générer des *hypergraphes de précédence* représentant ces mêmes séquences d'enchaînement.

La complexité de cette méthode est d'ordre polynômial, tout en étant plus élevé que celle de la méthode par transformation de graphes dans certains cas, il en reste tout de même que la deuxième méthode a généralement une complexité inférieure.

Nous remarquons que la deuxième méthode permet de générer aussi bien des graphes de précédence que des hypergraphes de précédence. Elle est plus facile à mettre en place que la méthode par transformation de graphes. Il est possible de dire que notre objectif est atteint, car nous avons proposé une méthode simple et efficace de génération de graphes de précédence.

[21] k est un entier strictement positif.

Après ces travaux, nous pourrions envisager leur programmation, et leur comparaison avec d'autres approches au niveau du temps de calcul.

Après cette programmation, il faudrait se pencher sur le découpage des hypergraphes de précédence afin de les utiliser en ALB. Pour cela deux solutions s'offrent à nous, la première est la décomposition des hypergraphes de précédence ainsi générés en graphes de précédence et l'utilisation de méthodes existantes, la deuxième est de proposer une méthode de découpage d'hypergraphes de précédence.

Pour la décomposition des hypergraphes en graphes de précédence, il suffit de transformer l'expression logique 2-STA, équivalente aux hypergraphes de précédence, en une expression 1-STA, équivalente à des graphes de précédence.

Pour le découpage des hypergraphes de précédence, nous pourrions peut être utiliser les méthodes d'équilibrage de lignes d'assemblage en considérant un hypergraphe comme un graphe de précédence avec des contraintes de précédence non fixes, c'est à dire pour une partie de l'hyperarc, nous gardons la contrainte de précédence, et pour l'autre partie, nous testons chaque cas possible.

Annexe A

Quelques généralités sur les graphes

L'ensemble des rappels proposé ici, porte sur la théorie des graphes. Cette annexe sera illustrée à l'aide de l'exemple de la figure A.1. Les définitions suivantes sont celles des *cocycles*, des *prédécesseurs — successeurs*, des *degrés* par rapport à un sommet et des *nombres d'arcs* par rapport à une paire de sommet d'un graphe donné, mais nous commencerons par des notions plus simples.

A.1 Notions de base

A.1.1 Graphe simple

Définition A.1 (Graphe simple)
Un graphe est dit simple si toute paire x, y de deux sommets distincts possède une arête commune et une seule.

C'est-à-dire, il existe l'arête (x, y) ou l'arête (y, x).

A.1.2 Graphe connexe

Définition A.2 (Graphe connexe)
Un graphe est dit connexe si pour toute paire x, y de deux sommets distincts, il existe une chaîne[1] *$\mu[x, y]$ reliant ces deux points.*

Tous les sommets d'un graphe connexe sont accessibles.

[1] Une chaîne est une succession d'arcs passant du sommet x au sommet y.

A.1.3 Chemin dans un graphe

Définition A.3 (Chemin)
Un chemin $\nu = (u_1, u_2, \ldots, u_q)$ est une succession de q arcs u_i tel que pour $i < q$, l'extrémité terminale de l'arc u_i coïncide avec l'extrémité initiale de l'arc u_{i+1}.

Si un chemina le même sommet initial et final, nous avons alors les deux possibilités suivantes.

A.1.3.1 Boucle dans un graphe

Définition A.4 (Boucle)
Une boucle $\nu = (u_1)$ est un arc tel que p l'extrémité terminale de l'arc coïncide avec l'extrémité initiale de lui-même.

C'est-à-dire qu'il existe au moins un chemin qui relie x à lui même en ne passant par aucun autre sommet.

A.1.3.2 Circuit dans un graphe

Définition A.5 (Circuit)
Un circuit $\nu = (u_1, u_2, \ldots, u_q)$ est une succession de q arcs u_i tel que pour $i < q$, l'extrémité terminale de l'arc u_i coïncide avec l'extrémité initiale de l'arc u_{i+1} et de plus l'extrémité terminale de l'arc u_q coïncide avec l'extrémité initiale de l'arc u_1.

C'est-à-dire qu'il existe au moins un chemin qui relie x à lui même en passant par au moins un autre sommet.

A.1.4 Chemin hamiltonien

Définition A.6 (Chemin hamiltonien)
Un chemin est dit hamiltonien s'il passe par tous les sommets d'un graphe, une et une seule fois.

A.1.5 Rang d'un graphe

Définition A.7 (Rang)
Nous appelons rang d'un graphe simple, sans circuit et orienté, le nombre maximal de nœuds entre le nœud racine et sa feuille la plus éloignée (en les incluant).

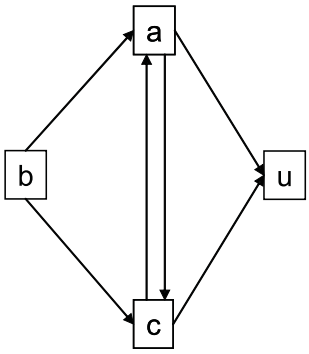


FIG. A.1 – Graphe orienté non-simple quelconque

A.2 Cocycle

DÉFINITION A.8 (COCYCLE)
Soit $G = (X, E)$ un graphe orienté. On appelle cocycle *d'un sommet l'ensemble des arcs qui lui sont incidents*[2]*. On note :*

$$\omega_G(i) = \{(i,j) \in E\} \cup \{(k,i) \in E\} \tag{A.1}$$

On appelle demi cocycle supérieur[3] *d'un sommet l'ensemble des arcs sortant de ce sommet. Et on appelle* demi cocycle inférieur[4] *d'un sommet l'ensemble des arcs entrant en ce sommet. Nous les notons :*

$$\begin{cases} \omega_G^+(i) = \{(i,j) \in E\} \text{ demi cocycle supérieur de } i \\ \omega_G^-(i) = \{(k,i) \in E\} \text{ demi cocycle inférieur de } i \end{cases} \tag{A.2}$$

Avec le graphe G de la figure A.1, nous avons $G = (X, E)$ où $X = \{a, b, c, d, u\}$ et $E = \{(a,b), (a,c), (c,b), (b,c), (c,d), (b,d), (d,b), (b,u), (d,u)\}$, et les cocycles supérieurs sont :

- $\omega_G^+(a) = \{(a,b), (a,c)\}$
- $\omega_G^+(b) = \{(b,c), (b,d), (b,u)\}$
- $\omega_G^+(c) = \{(c,b), (c,d)\}$

[2] Arcs entrant et arcs sortant de ce sommet.
[3] appelé aussi demi cocycle extérieur
[4] appelé aussi demi cocycle intérieur

- $\omega_G^+(d) = \{(d,b),(d,u)\}$
- $\omega_G^+(u) = \emptyset$

Les cocycles inférieurs du graphe G de la figure A.1, sont :

- $\omega_G^-(a) = \emptyset$
- $\omega_G^-(b) = \{(a,b),(c,b),(d,b)\}$
- $\omega_G^-(c) = \{(a,c),(b,c)\}$
- $\omega_G^-(d) = \{(b,d),(c,d)\}$
- $\omega_G^-(u) = \{(b,u),(d,u)\}$

A.3 Nombre d'arcs

Définition A.9 (Nombre d'arcs)
Soit $G = (X,E)$ un graphe orienté. On appelle nombre d'arcs *entre le sommet $i \in X$ et le sommet $j \in X$ d'un graphe G, les quantités d'arêtes attachées aux sommets et qui ont pour extrémité initiale le sommet i et pour extrémité terminale le sommet j, nous notons :*

$$m_G(i,j) = |\{(i,j) \in E : (i,j) \in \omega_G^+(i) \wedge (i,j) \in \omega_G^-(j)\}| \tag{A.3}$$

Le nombre d'arcs entre chaque paire de sommets d'un ASTD est :

$$\begin{array}{lllll}
m_G(a,a)=0 & m_G(a,b)=1 & m_G(a,c)=1 & m_G(a,d)=0 & m_G(a,u)=0 \\
m_G(b,a)=0 & m_G(b,b)=0 & m_G(b,c)=1 & m_G(b,d)=1 & m_G(b,u)=1 \\
m_G(c,a)=0 & m_G(c,b)=1 & m_G(c,c)=0 & m_G(c,d)=1 & m_G(c,u)=0 \\
m_G(d,a)=0 & m_G(d,b)=1 & m_G(d,c)=0 & m_G(d,d)=0 & m_G(d,u)=1 \\
m_G(u,a)=0 & m_G(u,b)=0 & m_G(u,c)=0 & m_G(u,d)=0 & m_G(u,u)=0
\end{array}$$

Afin de simplifier les notations, nous n'écrivons que les nombres d'arcs différents de zéro. Ce qui nous donne alors :

- $m_G(a,b) = 1$
- $m_G(a,c) = 1$
- $m_G(b,c) = 1$
- $m_G(b,d) = 1$
- $m_G(b,u) = 1$
- $m_G(c,b) = 1$
- $m_G(c,d) = 1$
- $m_G(d,b) = 1$
- $m_G(d,u) = 1$

Annexe B

Résolution des disjonctions d'après

Cette annexe traite de la résolution des disjonctions dans la génération des graphes de précédence d'après K.S. Naphade [42], [43], [44]. Cette résolution sera développée ci-après, et sera inspirée des remarques faites par P. De Lit dans l'annexe D de sa thèse [15]. Dans une dernière partie, nous essayerons d'apporter une solution à cette résolution.

B.1 Approche K.S. Naphade

Cette section présente la méthode de génération de graphes de précédence de K.S. Naphade et al. Ce travail est basé sur trois grandes étapes : la décomposition du problème en sous-problèmes, la représentation des sous-problèmes en graphes de décision et le partitionnement des graphes de décision. L'étape de décomposition du problème en sous-problèmes est la transformation d'un ensemble de contraintes de précédence en un ensemble de problèmes deux-satisfait (2-SAT).

Définition B.1 (k-SAT)
Clause de longueur inférieure ou égale à k et ne contenant pas d'opérateur «·»

Par exemple, voici un ensemble de clauses k-SAT :

$$\left.\begin{array}{c} x_1 + x_2 + \ldots + x_k \\ \vdots \\ x_{n-k} + \ldots + x_n \end{array}\right\}$$

La résolution d'un problème complet revient à résoudre l'ensemble des sous-problèmes engendrés en les modélisant par des graphes de décision. Après partitionnement, à chaque graphe de décision correspond un seul graphe de précédence d'assemblage.

B.1.1 La décomposition

Soit P un ensemble de N contraintes de précédence. Les contraintes de précédence de a sur b se notent «$\prec$». Les contraintes de précédence entre deux noeuds d'un graphe sont $a \prec b$, $b \prec a$, $a \equiv b$. Ces contraintes se traduisent respectivement par l'existence d'arcs entre les noeuds a et b $(a \rightarrow b)$, $(b \rightarrow a)$ ou la non existence d'arc entre les noeuds a et b. P est constitué de conjonctions de contraintes de précédence, ainsi que de disjonctions d'au moins deux contraintes de précédence. Dans le cas où une disjonction porte sur au moins trois contraintes de précédence, celle ci sera décomposée de telle manière que l'ensemble des Q_i soit 2-SAT.

Cet ensemble de N contraintes peut se découper en un ensemble de Q_i problèmes 2-SAT (voir l'équation (B.3)). Nous avons, pour toutes les expressions Q_i, un ensemble de disjonctions comportant seulement deux éléments et/ou un ensemble de conjonctions.

Nous allons considérer l'exemple suivant :

$$\begin{array}{ll} P: & 1 \prec 2 \\ & (1+4) \prec 5 \\ & (3+4+5) \prec 6 \\ & (3+5+6+7) \prec 9 \end{array} \tag{B.1}$$

Avec les équations (B.1), nous pouvons écrire les équations (B.2) en les décomposant en sous contraintes.

$$\begin{array}{l} ((3+4) \prec 6) + (5 \prec 6) \\ ((3+5) \prec 9) + ((6+7) \prec 9) \end{array} \tag{B.2}$$

Le problème P se décompose alors en quatre sous-problèmes :

$$\begin{array}{ll} Q_1: & (1 \prec 2) \wedge ((1+4) \prec 5) \wedge ((3+4) \prec 6) \wedge ((3+5) \prec 9) \\ Q_2: & (1 \prec 2) \wedge ((1+4) \prec 5) \wedge (5 \prec 6) \wedge ((3+5) \prec 9) \\ Q_3: & (1 \prec 2) \wedge ((1+4) \prec 5) \wedge ((3+4) \prec 6) \wedge ((6+7) \prec 9) \\ Q_4: & (1 \prec 2) \wedge ((1+4) \prec 5) \wedge (5 \prec 6) \wedge ((3+5) \prec 9) \end{array} \tag{B.3}$$

Comme nous l'avons vu ci-avant, une solution de P est une solution d'au moins une équation Q_i, et réciproquement, alors résoudre l'équation P revient à résoudre l'ensemble des équations Q_i

B.1.2 La représentation

Comme les clauses disjonctives de Q_i peuvent s'écrire en un ensemble de *si ... alors*.

$$\begin{aligned} (p+q) &\Leftrightarrow (\neg p \Rightarrow q) \\ &\Leftrightarrow (\neg q \Rightarrow p) \end{aligned} \tag{B.4}$$

Avec le deuxième terme de l'équation B.1 — $(1+4) \prec 5$ — et les équations B.4, nous pouvons alors écrire les équations B.5.

$$\begin{aligned} ((1 \prec 4) + (1 \prec 5)) &\Leftrightarrow (\neg(1 \prec 4) \Rightarrow (1 \prec 5)) \\ &\Leftrightarrow (\neg(1 \prec 5) \Rightarrow (1 \prec 4)) \end{aligned} \tag{B.5}$$

Suite à cette transformation de la disjonction en implication et l'apparition du non (noté $\neg$), l'expression $a \equiv b$ est introduite pour traduire l'absence de contraintes de précédence entre les conditions a et b. Alors l'expression (B.5) peut s'écrire sous la forme suivante :

$$\begin{array}{l} (4 \prec 1) \Rightarrow (1 \prec 5) \\ (1 \equiv 4) \Rightarrow (1 \prec 5) \\ (5 \prec 1) \Rightarrow (1 \prec 4) \\ (5 \equiv 1) \Rightarrow (1 \prec 4) \end{array} \tag{B.6}$$

Une nouvelle notation $\Re$ a alors été introduite afin de traduire les deux expressions $(a \prec b)$ et $(a \equiv b)$ en une seule $(a \Re b)$. L'expression (B.6) devient alors :

$$\begin{array}{l} (4 \Re 1) \Rightarrow (1 \prec 5) \\ (5 \Re 1) \Rightarrow (1 \prec 4) \end{array} \tag{B.7}$$

A l'aide de l'expression (B.7), le graphe de décision G^* est généré. Pour cette génération, une notion est introduite. Supposons que nous appelons A la contrainte de

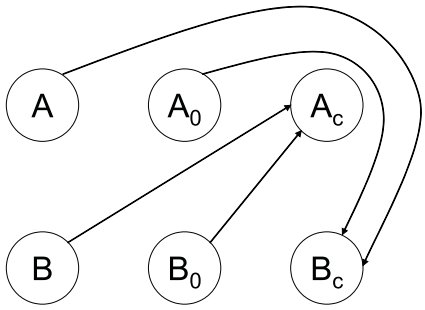


FIG. B.1 – Graphe de décision G^* associé aux contraintes de précédence $((1+4) \prec 5)$

précédence $(4 \prec 1)$, nous appelons A_c la contrainte de précédence $(1 \prec 4)$ et A_0 la contrainte $(4 \equiv 1)$. Nous faisons de même avec $(5 \prec 1)$ que nous nommons B alors $(1 \prec 5)$ s'écrit B_c et $(5 \equiv 1)$ s'écrit B_0. D'après l'expression (B.7), nous avons $A \Rightarrow B_c$, $A_0 \Rightarrow B_c$, $B \Rightarrow A_c$, $B_0 \Rightarrow A_c$. Cette écriture peut alors être représentée par le graphe de décision de la figure B.1.

B.1.3 Le partitionnement

K.S. NAPHADE appelle l'ensemble C_I le complémentaire de l'ensemble I des contraintes de précédence. —$C_I = \{I_0, I_c\}$ ou encore $C_{I_0} = \{I, I_c\}$ ou encore $C_{I_c} = \{I, I_0\}$ —

PROPRIÉTÉ B.1 (PARTITIONNEMENT)
La faisabilité et la consistance de la partition de G^* dans R et A doivent satisfaire aux propriétés suivantes :

Répartition : *Pour l'ensemble des noeuds $\{I, I_0, I_c\}$, un et un seul des noeud doit appartenir à A, les deux autres doivent appartenir à R.*

Appartenance : *Il n'existe pas d'arc (I, J) appartenant à G^* où I appartient à A, et J appartient à R.*

La propriété de *répartition* assure l'acceptation d'une et d'une seule contrainte de précédence à l'ensemble A, et la propriété d'*appartenance* assure elle, l'acceptation de J si I est accepté et si I implique J.

DÉFINITION B.2 (PARTITIONNEMENT)
Le problème de génération de graphe de précédence repose sur le partitionnement des graphes de décision en deux ensembles A et R, avec une et une seule alternative pour chaque décision appartenant à A, et l'orientation des arcs de R vers A.

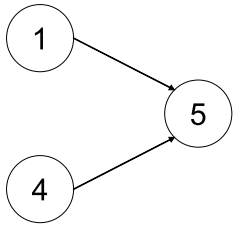


FIG. B.2 – Graphe de précédence possible G associé à la contrainte $((1+4) \prec 5)$

Avec l'expression (B.6), la propriété B.1 et la définition B.2, nous obtenons les ensembles $A = \{A_0, B_c\}$ et $R = \{A, A_c, B, B_0\}$. Avec ce partitionnement, nous obtenons un graphe de précédence possible, présenté par la figure B.2.

B.1.4 Évaluation

Cette méthode possède l'avantage de permettre de générer toutes les partitions possibles de G^* avec la décomposition du problème P en un ensemble de sous-problèmes Q_i. Cette approche est une piste à explorer, mais actuellement elle n'est pas valide, car elle donne dans quelques cas particuliers des « graphes traduisant les précédences » entre les tâches qui ne peuvent pas être appelés des graphes de précédence car ils sont cycliques — voir figure B.3 — avec le problème P suivant :

$$\begin{aligned} P: \quad & (1+2) \prec 0 \\ & (1 \prec (2+3) \end{aligned} \tag{B.8}$$

Après une suite de transformations respectant la méthode de K.S. NAPHADE, nous obtenons par exemple :

$$\begin{aligned} B &\Rightarrow A_c \\ C &\Rightarrow D_c \\ (0 \prec 2) &\Rightarrow (1 \prec 0) \\ (2 \prec 1) &\Rightarrow (1 \prec 3) \end{aligned} \tag{B.9}$$

Il est possible de représenter ce graphe par la figure B.3.

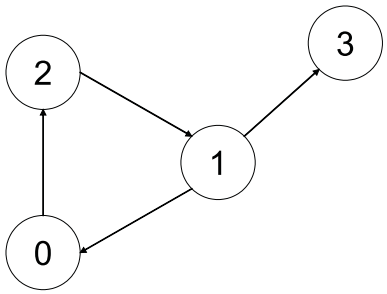


FIG. B.3 – Graphe « de précédence » invalide

B.2 Approche P. De Lit

P. De Lit a repris l'approche de K.S. Naphade afin de l'améliorer, pour ce faire, il a remplacé les opérateurs $\equiv$ et $\Re$ par l'opérateur $\preceq$. Cet opérateur, appliqué à a et b de la manière suivante $a \preceq b$ traduit l'impossibilité de réaliser b avant a. L'opérateur $\preceq$ est distributif à droite et à gauche par rapport au « et » comme par rapport au « ou ». P. De Lit a introduit un ensemble de définitions — voir [15] —, sans toutefois résoudre les problèmes de génération de graphes invalides. Nous remarquerons tout de même que les graphes générés par cette approche sont valides, ce sont des graphes de précédence, mais ils engendrent des séquences non valides.

Si nous reprenons l'exemple donné par P. De Lit [15], nous obtenons, avec les contraintes $(d+e) \prec f$ et $f \prec (d+e)$, les quatre graphes de précédence suivants :
$G_1 =< \{d,e,f\}, \{(d,f),(e,f)\} >$, $G_2 =< \{d,e,f\}, \{(e,f),(f,d)\} >$,
$G_3 =< \{d,e,f\}, \{(d,f),(f,e)\} >$, $G_4 =< \{d,e,f\}, \{(f,d),(f,e)\} >$.

Nous pouvons voir que les graphes G_1 et G_2 ne respectent pas l'ensemble de contraintes proposées au départ. Il y a un problème de résolution des disjonctions des contraintes de précédence comme le dit P. De Lit. Suite à ces travaux, nous proposons l'amélioration suivante.

B.3 Approche K.S. Naphade – P. De Lit améliorée

Notre proposition est relativement simple, nous reprenons l'ensemble de la méthode de K.S. Naphade, juste en modifiant le complémentaire de $a \prec b$, (qui est, dans l'approche de K.S. Naphade : $b \equiv a$ et dans celle de P. De Lit : $b \preceq a$) par $a \succ b$.

Supposons que les graphes d'assemblage soient linéaires, nous avons donc une seule tâche en même temps, alors a précède b ou alors a succède b. Donc le complémentaire de $a \prec b$ est $a \succ b$. Maintenant supposons que les graphes d'assemblage ne soient pas linéaires, alors si les tâches a et b ne sont pas sur la même *branche* de la ligne d'assemblage, nous pouvons considérer que ces deux branches forment chacune un système différent, c'est à dire que nous considérons le sous-assemblage sortant d'une chaîne d'assemblage comme un produit fini pour cette chaîne qu'il quitte, et comme un composant élémentaire pour la chaîne dans laquelle il rentre. Nous allons réécrire les propriétés et définitions qui le méritent.

Nous appelons l'ensemble C_I le complémentaire de l'ensemble I des contraintes de précédence. — $C_I = \{I_c\}$ ou encore $C_{I_c} = \{I\}$ —

Propriété B.2 (Partitionnement amélioré)
La faisabilité et la consistance de la partition de G^* dans R et A doivent satisfaire les propriétés suivantes :

Répartition : *Pour l'ensemble des noeuds $\{I, I_c\}$, un et un seul des nœuds doit appartenir à A, l'autre doit appartenir à R.*

Appartenance : *Il n'existe pas d'arc (I, J) appartenant à G^* où I appartient à A et J appartient à R.*

La propriété de *répartition* assure l'acceptation d'une et d'une seule contrainte de précédence à l'ensemble A, et la propriété d'*appartenance* assure elle, l'acceptation de J si I est accepté et si I implique J.

Définition B.3 (Partitionnement amélioré)
Le problème de génération des graphes de précédence repose sur le partitionnement des graphes de décision en deux ensembles A et R, avec une et une seule alternative pour chaque décision appartenant à A et une et une seule alternative pour chaque décision appartenant à R, et l'orientation des arcs de R vers A.

Reprenons l'exemple de P. De Lit :

$$\begin{array}{rcl} (d+e) & \prec & f \\ f & \prec & (d+e) \end{array} \tag{B.10}$$

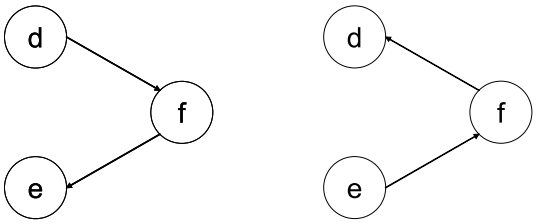


FIG. B.4 – Graphes de précédence valides associés aux contraintes de précédence de P. DE LIT [15]

Nous obtenons donc avec la propriété B.2 et la définition B.3 l'expression suivante :

$$\begin{aligned} &(d \succ f) \Rightarrow (e \prec f) \\ &(e \succ f) \Rightarrow (d \prec f) \\ &(f \succ d) \Rightarrow (f \prec e) \\ &(f \succ e) \Rightarrow (f \prec d) \end{aligned} \tag{B.11}$$

Nous obtenons alors les graphes de précédence $G_1 =< \{d, e, f\}, \{(e, f), (f, d)\} >$, $G_2 =< \{d, e, f\}, \{(d, f), (f, e)\} >$ — voir figure B.4 — représentant l'ensemble des séquences vérifiant les contraintes de précédence nommées ci-avant.

Annexe C

Les différentes étapes de simplification pour la 1-STA

Nous trouverons dans cette annexe, les différentes étapes de résolution entre l'équation 4.21 et l'équation 4.31 (rappelée C.16).

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot((\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.1}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot((\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \\
& \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \\
& u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \\
& u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to \\
& u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \\
& \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \\
& u)\cdot(\alpha_4 \to u) + (\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.2}
$$

$$
\begin{aligned}
\mathcal{L}_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot((\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + \\
& (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.3}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot((\alpha \to \\
& u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot \\
& (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.4}
$$

$$\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot((\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \\
& u)\cdot(\alpha_4 \to u) + (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \\
& u) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned} \tag{C.5}$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot((\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) \\
& + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \\
& u))
\end{aligned}
\tag{C.6}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot((\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_4)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + \\
& (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_2)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.7}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot((\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_4 \to u) + \\
& (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_4 \to u) + \\
& (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_4 \to u) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to u) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to u) + \\
& (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to u) + \\
& (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to u) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to u))
\end{aligned}
\tag{C.8}
$$

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot((\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\ & \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\ & \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_1 \to \\ & \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + \\ & (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3) \\ & + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\ & \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\ & \alpha_1)\cdot(\alpha_3 \to \alpha_1) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \\ & \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\ & \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\ & \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \\ & \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3) + \\ & (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) \\ & + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \\ & \alpha_1) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \\ & \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \\ & \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1))\end{aligned} \tag{C.9}$$

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \\ & \alpha_1))\cdot((\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4) + (\alpha_1 \to \alpha_3)\cdot \\ & (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_2 \to \alpha_3) + (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \\ & \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3) + \\ & (\alpha_2 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \\ & \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \\ & \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3) + \\ & (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_3 \to \alpha_2) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\ & \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_3 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \\ & \alpha_1)\cdot(\alpha_3 \to \alpha_2) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)) \\ & + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\\ & (\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\ & \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_2 \to \alpha_1))\end{aligned} \tag{C.10}$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \\
& \alpha_1))\cdot((\alpha_1 \to \alpha_3)+(\alpha_3 \to \alpha_1))\cdot((\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4) \\
& + (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_2 \to \alpha_3) + (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4) + (\alpha_2 \to \\
& \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) \\
& + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to u)\cdot \\
& (\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\\
& (\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \\
& \alpha_1))
\end{aligned}
\tag{C.11}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot((\alpha_1 \to \alpha_2) + \\
& (\alpha_2 \to \alpha_1))\cdot((\alpha_1 \to \alpha_3) + (\alpha_3 \to \alpha_1))\cdot((\alpha_2 \to \alpha_3) + (\alpha_3 \to \alpha_2)) \\
& \cdot((\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_4) + (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2) + (\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_4 \to \alpha_2) + (\alpha_2 \to \alpha_4)\cdot(\alpha_1 \to \alpha_4) + (\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2) + (\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\ (\alpha_3 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_3 \to \alpha_2)\cdot(\alpha_2 \to \alpha_1))
\end{aligned}
\tag{C.12}
$$

$$
\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \\
& \alpha_1))\cdot((\alpha_1 \to \alpha_3)+(\alpha_3 \to \alpha_1))\cdot((\alpha_2 \to \alpha_3)+(\alpha_3 \to \alpha_2))\cdot((\alpha_1 \to \\
& \alpha_4)+(\alpha_4 \to \alpha_1))\cdot((\alpha_2 \to \alpha_4) + ((\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2) + \\
& (\alpha_2 \to \alpha_4) + (\alpha_2 \to \alpha_4) + (\alpha_2 \to \alpha_4) + (\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2) \\
& + (\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2) + (\alpha_4 \to \alpha_2)) \\
& + (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\ (\alpha_1 \to \alpha_2) + (\alpha_2 \to \alpha_1))
\end{aligned}
\tag{C.13}
$$

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \\ & \alpha_1))\cdot((\alpha_1 \to \alpha_3)+(\alpha_3 \to \alpha_1))\cdot((\alpha_2 \to \alpha_3)+(\alpha_3 \to \alpha_2))\cdot((\alpha_1 \to \\ & \alpha_4)+(\alpha_4 \to \alpha_1))\cdot((\alpha_2 \to \alpha_4)+(\alpha_4 \to \alpha_2)) \\ & +\ (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\ & \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\end{aligned} \tag{C.14}$$

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\ & u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot(\alpha_4 \to \alpha_3) \\ & +\ (\alpha \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\end{aligned} \tag{C.15}$$

$$\begin{aligned}\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \\ & u)\cdot(\alpha_4 \to \alpha_3) \\ & +\ (\alpha \to \alpha_3)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\end{aligned} \tag{C.16}$$

Annexe D

Les différentes étapes de simplification pour la 2-STA

Nous trouverons dans cette annexe, les différentes étapes de résolution entre l'équation 4.21 et l'équation 4.32 (rappelée D.6). Pour cela, nous utilisons les équations de (C.1) à (C.9), puis nous avons l'ensemble des équations suivantes.

$$\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to \\
& u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot((\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3) + (\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \\
& \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_1 \to \\
& \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + \\
& (\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \\
& \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_3 \to \alpha_1) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \\
& \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \\
& \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3) + \\
& (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) \\
& + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_1) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_1 \to \alpha_2) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \\
& \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_2 \to \alpha_1))
\end{aligned} \tag{D.1}$$

$$\begin{aligned}
\pounds_{\Upsilon}^{\mathcal{D}v} \equiv\ & (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot \\
& (\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot((\alpha_1 \to \alpha_2) + (\alpha_2 \to \\
& \alpha_1))\cdot((\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3) \\
& + (\alpha_1 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) \\
& + (\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) \\
& + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_1) \\
& + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2) \\
& + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) \\
& + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_3) \\
& + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) \\
& + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_3 \to \alpha_1) \\
& + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2) + \\
& (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_3 \to \alpha_1))
\end{aligned} \tag{D.2}$$

$$\pounds_{\Upsilon}^{\mathcal{D}v} \equiv (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \alpha_1))\cdot((\alpha_1 \to \alpha_3)+(\alpha_3 \to \alpha_1))\cdot((\alpha_1 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3) + (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_1 \to \alpha_4)\cdot(\alpha_4 \to \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_2 \to \alpha_4)\cdot(\alpha_2 \to \alpha_3)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2) + (\alpha_3 \to \alpha_4)\cdot(\alpha_3 \to \alpha_2)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_2 \to \alpha_3) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_3 \to \alpha_2) + (\alpha_4 \to \alpha_3)\cdot(\alpha_4 \to \alpha_2)\cdot(\alpha_4 \to \alpha_1)\cdot(\alpha_3 \to \alpha_2)) \tag{D.3}$$

$$\pounds_{\Upsilon}^{\mathcal{D}v} \equiv (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot((\alpha_1 \to \alpha_2)+(\alpha_2 \to \alpha_1))\cdot((\alpha_1 \to \alpha_3) + (\alpha_3 \to \alpha_1))\cdot((\alpha_4 \to \alpha_3) + ((\alpha_4 \to \alpha_1)\cdot(\alpha_4 \to \alpha_2))) \tag{D.4}$$

$$\pounds_{\Upsilon}^{\mathcal{D}v} \equiv (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha \to u)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot(\alpha_4 \to u)\cdot\cdot((\alpha_4 \to \alpha_3)+(\alpha_4 \to \alpha_1))\cdot((\alpha_4 \to \alpha_3) + (\alpha_4 \to \alpha_1)) \tag{D.5}$$

$$\pounds_{\Upsilon}^{\mathcal{D}v} \equiv (\alpha \to \alpha_1)\cdot(\alpha \to \alpha_2)\cdot(\alpha \to \alpha_3)\cdot(\alpha \to \alpha_4)\cdot(\alpha_1 \to u)\cdot(\alpha_2 \to u)\cdot(\alpha_3 \to u)\cdot\cdot((\alpha_4 \to \alpha_3)+(\alpha_4 \to \alpha_1))\cdot((\alpha_4 \to \alpha_3)+(\alpha_4 \to \alpha_1)) \tag{D.6}$$

Bibliographie

[1] E. Bampis, J. C. Konig, and D. Trystram. Minimizing the schedule length for a parallel 3d-grid precedence graph. In *European journal of Operational Research 95*, pages 427–438, 1996.

[2] P. Baptiste, C. H. Cho, J. Favrel, and M. Zouhri. Une caractérisation analytique des ordonnancements admissibles sous contraintes hétérogènes en flow-shop. In *APII*, volume 25, pages 87–102, 1991.

[3] I. Baybars. On currently practised formulations of the assembly line balancing problem. In *Journal of Operations Management*, pages 449–453, 1985.

[4] C. Berge. *Graphes et Hypergraphes*, volume 37. Dunod, 2ième edition, 1967.

[5] N. Boneschanscher. *Plan Generation for Flexible Assembly Systems.* Thèse de doctorat, Delft University of Technology, Delft, The Netherlands, 1993.

[6] N. Boneschanscher, J. H. M. van der Drift, S. J. Buckley, and R. H. Taylor. Subassembly stability. In *AAAI, 7th National Conference on Artificial Intelligence, St. Paul, MN*, August 21-26 1988.

[7] N. Boneschanscher and C. J. M. Heemskerk. Grouping parts to reduce the complexity of assembly sequence planning. In *INCOM'89, Madrid*, pages 26 – 29, September 1989.

[8] K. S. Booth. Testing for the consecutive ones property graphs, and graphplanarity using pq-tree algorithms. In *Journal of Computer and System Science 143*, pages 335–379, 1984.

[9] G. Boothroyd and P. Dewhurst. Design for assembly hanbook. In *Departement of Mechznical Engineering University of Massachussets*, 1983.

[10] A. Bourjault. *Contribution à une approche méthodologique de l'assemblage automatisé : Elaboration automatique des séquences opératoires.* Thesis to obtain grade de docteur es sciences physiques, Université de Franche-Comté, No. 188, Nov. 1984.

[11] A. Bratcu. *Détermination systématique des graphes de précédence et équilibrage des lignes d'assemblage.* Thèse de doctorat, Université de Franche-Comté, No. 860, July. 2001.

[12] K. Chen. *Contribution à une méthode systématique de détermination des graphes de précédence pour l'assemblage.* Thèse de doctorat, Université de Franche-Comté, No. 520, July. 1996.

[13] J. Danloy, F. Petit, A. Leroy, P. De Lit, and B. Rekiek. A pragmatic approach for precedence graphs generation. In *IEEE International Symposium on Assembly and Task Planning, ISATP'99*, pages 387–392, Porto, Portugal, July 1999.

[14] T. L. De Fazio and D. E. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE JRA*, RA-3(6), 1987.

[15] P. De Lit. *A comprehensive and integreted approach for the design of product familly and its assembly system.* Docteur en sciences appliquées, Université libre de Bruxelles, Belgique, 2000.

[16] A. Delchambre. *Conception assistée par ordinateur des gammes opératoires d'usinage.* Ph. d. thesis, Université Libre de Bruxelles, 1990.

[17] A. Delchambre. A pragmatic approach to computer-aided assembly planning. In *1990 IEEE International Conference on Robotics and Automation*, pages 1600–1605, Cincinnati, Ohio, USA, IEEE Computer Society Press, 1990.

[18] A. Delchambre. *Computer-aided assembly planning.* Chapman and Hall, London, United Kingdom, first edition, 1992.

[19] G. Dini and M. Santochi. Automated sequencing and subassembly detection in assembly planning. In *Annals of the CIRP*, pages 1–4, 14/1/1992.

[20] A. Enmer. *Équilibrage des chaînes d'assemblage : bibliographie et proposition d'une méthode permettant le parallélisme.* Mémoire de dea, INSA, Lyon, France, 1989.

[21] P. Fouda. Génération de gammes d'assemblage pour les familles de produits. In *Travail de spécialisation présenté dans le cadre du DES en productique*, Université Libre de Bruxelles, Belgique, 1999.

[22] P. Fouda, P. De Lit, J. Danloy, B. Rekiek, T. L'Eglise, and A. Delchambre. A heuristic to generate a precedence graph between components for a product family. In *ISATP 2001*, volume M1B-3, pages 43–48, 2001.

[23] P. Fouda, P. De Lit, T. L'Eglise, B. Rekiek, and A. Delchambre. Une heuristique pour la génération du graphe de précédence d'une famille de produit. In *MOSIM'01*, pages 127–134, April 2001, Troyes.

[24] P. Fouda, P. De Lit, B. Rekiek, and A. Delchambre. Generation of precedence graphs for a product family using a disassembly approach. In *ISATP 2001*, T1B-4, pages 226–231, 2001.

[25] B. Frommherz and J. Hornberger. Automatic generation of precedence graphs. In *Proc. Int. Symposium on Industrial Robotics, Lausanne, Switzerland*, pages 453–466, April 1988.

[26] S. Ghosh and R. J. Gagnon. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. In *Int. journal of Production Research*, volume 12-4, pages 637–670, 1989.

[27] C. J. M. Heemskerk. The assembly state transition diagram, a representation for assembly sequences. In *Technical Report WPS-88. 048, Laboratory for Manufacturing Systems, Delft University of Technology, Delft, The Netherlands*, 1988.

[28] C. J. M. Heemskerk. The use of heuristics in assembly sequence planning. In *Annals of the CIRP*, volume 38/1/1989, pages 37–40, Jannuary 1989.

[29] C. J. M. Heemskerk. *A Concept for Computer Aided Process Planning for Flexible Assembly.* Thèse de doctorat, Delft University of Technology, Delft, The Netherlands, ISBN 90-370-0041X, 1990.

[30] J. M. Henrioud. *Contribution à la conceptualisation de l'assemblage automatisé : nouvelle approche en vue de la détermination des processus d'assemblage.* Thèse de doctorat, Université de Franche-Comté, France, 1989.

[31] J. M. Henrioud. Lega : A efficient computer-aided generator of assembly plans. In *FAIM91*, mars 1991.

[32] J. M. Henrioud and A. Bourjault. Lega : A computer-aided generator of assembly plans. In *Chapter 8 in : Computer-aided mechanical assembly planning. Ed. L. S. Hommen de Mello, S. Lee, Klüwer Academic publishers*, pages 191–215, 1991.

[33] J. M. Henrioud and A. Bourjault. Computer aided assembly process planning. In *journal of Engineering Manufacture*, volume 206, pages 61–66, 1992.

[34] J. M. Henrioud and A. Bratcu. Algorithm for generating the precedence graphs in assembly systems. In *CSCC'99, Athens*, 1-555, July 1999.

[35] L. S. Homem de Mello and A. C. Sanderson. Representations of assembly sequences. In *Proc 11th IJCAI*, pages 1035–1040, Aug. 1989.

[36] F. Lhote. *Cours d'automatique générale Tome 1 : systèmes logiques combinatoires.* École Nationale Supérieure de Chronométrie et de Micromécanique de Besançon.

[37] T. I. Liu and G. Yuan. Pdes-an expert system for constructing the precedence diagram. In *J. Franklin Inst.*, volume 333B-6, pages 975–990, 1996.

[38] K. Mawussi. *Modèle de représentation et de définition d'outillages de forme complexe. Application à la génération automatique de processus d'usinage.* Thèse de doctorat, École nationale supérieure de Cachan, Janvier 1995.

[39] K. Mawussi and A. Bernard. Complex dies representation and definition with complex feature-based model. In *Proceedings of the Computers in Engineering Conference and Engineering Database Synposium ASME 1995*, pages 767–778, 1995.

[40] V. Minzu and A. Bratcu. Precedence graphs generation using assembly sequences. In *CSCC'99, Athens*, 1-556, July 1999.

[41] V. Minzu and J. M. Henrioud. Systematic method for the disign of flexible assembly systems. In *IEEE*, pages 56–62, 1993.

[42] K. S. Naphade. *A Graph Theoretic Framework for Integrated Assembly Planning.* Thèse de doctorat, Department of Industrial and Manufacturing Systems Engineering,Lehigh University, June 1997.

[43] K. S. Naphade, Robert H. Storer, and S. David Wu. Graph-theoric generation of assembly plans parts i :correct generation of precedence graphs. In *IMSE Technical Report 99T-003, Lehigh University, Pennsylvania*, 1999.

[44] K. S. Naphade, Robert H. Storer, and S. David Wu. Graph-theoric generation of assembly plans parts ii :problem decomposition and optimization algorithms. In *IMSE Technical Report 99T-003, Lehigh University, Pennsylvania*, 1999.

[45] C. Perrard. *Contribution à une méthodologie d'aide à l'implantation et à la spécification des équipements des systèmes flexibles d'assemblage.* Thèse de doctorat, Université de Franche-Comté, France, 1992.

[46] T. O. Prenting and R. M. Battaglin. The precedence diagram : a tool for analysis in assembly line balancing. *J. Ind. Eng.*, 15(4) :208–213, 1964.

[47] M. Queysanne. *Algèbre M. P. et spécialités AA.* COLLECTION U. SÉRIE MATHÉMATIQUES, armand collin edition, 1964.

[48] L. Relange and J. M. Henrioud. Systematic determination of assembly state transition diagrams. In *ISATP 2001*, M1B-5, pages 55–60, May, 2001.

[49] L. Relange and J. M. Henrioud. Systematic determination of assembly precedence graphs. In *Mecatronics 2001*, pages 495–500, Sept. , 2001.

[50] G. Vallet. *Équilibrage des lignes d'assemblage : étude bibliographique et présentation d'une méthode basée sur l'étude des chronogrammes.* Mémoire de dea, Université de Franche-Comté, France, 1987.

[51] J. Vélu. *Méthodes mathématiques pour l'informatique*. Dunod, 1994.

Index

RÉSUMÉ : Après une rapide présentation des systèmes d'assemblage et des différentes représentations des processus d'assemblage, ce travail de recherche présente plus précisément trois grands types de modélisation : les *graphes d'assemblage*, les *graphes de précédence* et les *ASTD*. Les graphes de précédence étant très utilisés avec les méthodes d'équilibrage ou de conception des lignes d'assemblage, l'objectif de ce travail est de proposer une méthode de génération des graphes de précédence simple et efficace à partir d'un ensemble de graphes d'assemblage préalablement établis. Deux méthodes de génération de graphe de précédence sont proposées dans ce travail : une *par transformation de graphes* et une directement basée sur la *logique booléenne*. La méthode par transformation de graphes permet d'obtenir un graphe de précédence si l'ensemble des séquences d'enchaînement peut être représenté par un unique graphe de précédence. Dans le cas contraire, avec les améliorations apportées à la méthode, il est possible d'obtenir soit un ensemble de graphes de précédence soit un hypergraphe de précédence. La deuxième de ces méthodes permet d'obtenir directement un ensemble de graphes de précédence ou d'hypergraphes de précédence selon le niveau de complexité du problème. Un calcul des complexités des algorithmes respectifs montre qu'ils sont polynomiaux.

MOTS-CLÉS : assemblage, graphe de précédence, hypergraphe, logique.

ABSTRACT : After a short introduction to the assembly systems and different representations of assembly process, this work presents three types of models : *assembly graphs*, *precedence graphs* and *ASTD*. The precedence graphs being currently used by assembly line balancing methods in order to assembly systems determination, the objective of this work is to propose an easy and effective method of precedence graphs generation from a set of assembly sequences. Two methods of generation of precedence graphs are presented here : the first one based on graph transformations and the second one based directly boolean logic. The graph transformation method gives a precedence graph if the set of the assembly sequences can be represented by one and only one precedence graph. In the opposite case, with some ameliorations, it is possible to obtain either a set of precedence graphs or a single precedence hypergraph. With the second method, we can obtain directly a set of precedence graphs or of precedence hypergraphs depending upon the set of assembly sequences. The evaluation of complexity of respective algorithms allows to say that they are polynomial.

KEYWORDS : assembly, precedence graph, hypergraph, boolean logic.